# Оглавление

## Две лягушки

Две лягушки на позициях $a$ и $b$ в ряду из $n$ кувшинок. Ходят по очереди (начинает Алиса), прыгая на соседнюю свободную кувшинку. Если ход невозможен — проигрывает. Оба играют оптимально. Определить, может ли Алиса победить.

---

Ключевое наблюдение: ответ зависит только от чётности расстояния между лягушками.

Каждый ход изменяет расстояние между лягушками на 1 и четность расстояния меняется каждый ход. Алиса ходит первой, поэтому если начальное расстояние четное, она всегда будет оставлять Бобу позицию с нечетным расстоянием. В проигрышной позиции (когда нельзя сделать ход) расстояние всегда нечетно. Следовательно, последний ход сделает Боб, и Алиса выиграет. Вычислим расстояние $d = |a - b|$. Если $d$ четно, Алиса выигрывает (YES). Если $d$ нечетно, Алиса проигрывает (NO).

## Таблица MEX

Дана таблица $n \times m$, в которую нужно расставить числа $0, 1, \ldots, n \cdot m - 1$ (каждое ровно один раз). Для строки или столбца mex — это минимальное неотрицательное число, отсутствующее в ней. Требуется максимизировать сумму mex по всем строкам и столбцам.

---

Заметим, что число 0 встречается ровно в одной строке и ровно в одном столбце. Число 1, если оно существует, может стоять только в строке или столбце, где есть 0. Поэтому сумма mex не может превышать $\max(n, m) + 1$, так как только в одном столбце (или строке) mex может быть больше 1, и ещё в одном может быть равен 1.

Пример достижения этой оценки: если $n > m$, то в первом столбце расставим числа от 0 до $n-1$ сверху вниз, а остальные элементы заполним произвольно. Тогда:

- В первом столбце mex $= n$ (содержит $0, 1, \ldots, n - 1$)
- В первой строке mex $= 1$ (содержит 0)
- Во всех остальных строках и столбцах mex $= 0$

Сумма: $n + 1 = \max(n, m) + 1$. Аналогично, если $m \geq n$, заполняем первую строку числами от 0 до $m - 1$. Таким образом, максимальная сумма равна $\max(n, m) + 1$.

## Соседние суммы цифр - легкая версия

Даны целые числа $x$ и $y$. Определить, существует ли положительное целое $n$, такое что сумма цифр $n$ равна $x$, а сумма цифр $n + 1$ равна $y$. Здесь $x, y \leq 1000$.

---

Заметим, что при переходе от $n$ к $n + 1$ сумма цифр изменяется определённым образом. Если $n$ не заканчивается на цифру 9, то $n + 1$ получается простым увеличением последней цифры на 1, поэтому сумма цифр увеличивается на 1. Значит, в этом случае $y = x + 1$.

Если же $n$ заканчивается на одну или несколько девяток, то происходит перенос. Пусть $n$ заканчивается на $k$ девяток ($k \geq 1$). Тогда:

1. $k$ девяток обнуляются, что уменьшает сумму цифр на $9k$
2. Цифра перед девятками увеличивается на 1, что увеличивает сумму на 1
3. Все остальные цифры не меняются

Итого: $y = x - 9k + 1$, или $x - y = 9k - 1$. Таким образом, возможны два типа пар $(x, y)$: $y = x + 1$ (случай без переноса) и $x - y = 9k - 1$ для некоторого $k \geq 1$ (случай с переносом), что эквивалентно $x - y \equiv 8 \pmod 9$ и $x > y$

Так как ограничения на $x$ и $y$ небольшие ($\leq 1000$), можно заранее вычислить все возможные пары $(x, y)$, которые могут быть получены как сумма цифр соседних чисел $n$ и $n + 1$.

Для этого рассматриваем два случая - без переноса: $y = x + 1$, таких пар существует 999 (при $x =$ 1..999); с переносом: $x = y + 9k - 1$ - здесь для фиксированных $y = 1, 2, 3, \ldots 1000$, и $k = 1, 2, 3, \ldots 1000$ вычисляем $x = y + 9k - 1$ и добавляем пару $(x, y)$ в множество допустимых пар.

После такого предподсчёта мы получаем полный список всех пар $(x, y)$, для которых существует число $n$ с требуемым свойством. Теперь для каждого запроса достаточно просто проверить, содержится ли заданная пара в этом предподсчитанном множестве.

## Соседние суммы цифр - сложная версия

Даны целые числа $x$ и $y$. Определить, существует ли положительное целое $n$, такое что сумма цифр $n$ равна $x$, а сумма цифр $n+1$ равна $y$. Здесь $x, y \leq 10^9$.

---

При переходе от $n$ к $n+1$ возможны два случая:

1. Без переноса: если $n$ не оканчивается на 9, то сумма цифр увеличивается на 1: $y = x + 1$

2. С переносом: если $n$ оканчивается на $k$ девяток ($k \geq 1$), то сумма уменьшается на $9k$ и увеличивается на 1: $y = x - 9k + 1$, что эквивалентно $x - y = 9k - 1$

Таким образом, пара $(x, y)$ достижима тогда и только тогда, когда выполняется одно из условий: $y = x + 1$ или $x > y$ и $(x - y + 1)$ делится на 9 (так как $x - y + 1 = 9k$)

## Разделение - легкая версия

Дана последовательность $a$ длины $2n$. Нужно разделить её на две подпоследовательности $p$ и $q$ длины $n$ каждая, чтобы максимизировать сумму $f(p) + f(q)$, где $f(b)$ - количество различных элементов, встречающихся в $b$ нечётное число раз. Здесь $n \leq 10$.

---

Поскольку $n \leq 10$ и сумма $n$ по всем тестам не превышает 100, можно использовать полный перебор. Будем перебирать все возможные разбиения массива $a$ на две группы по $n$ элементов. Для каждого разбиения вычислим $f(p) + f(q)$ и возьмём максимум.

## Разделение - сложная версия

Дана последовательность $a$ длины $2n$. Нужно разделить её на две подпоследовательности $p$ и $q$ длины $n$ каждая, чтобы максимизировать сумму $f(p) + f(q)$, где $f(b)$ — количество различных элементов, встречающихся в $b$ нечётное число раз. Здесь $n \leq 2 \cdot 10^5$.

---

Рассмотрим произвольное число, которое встречается в исходной последовательности $cnt$ раз. Пусть в $p$ оно попадает $u$ раз, а в $q$ — $v$ раз, где $u + v = cnt$. Вклад этого числа в сумму $f(p) + f(q)$ равен $(u \bmod 2) + (v \bmod 2)$, то есть может быть 0, 1 или 2.

Разделим все различные значения на три категории в зависимости от остатка $cnt$ по модулю 4:

- $x$ — количество значений, у которых $cnt$ нечётно ($cnt \equiv 1$ или $3 \pmod 4$)
- $y$ — количество значений с $cnt \equiv 2 \pmod 4$
- $z$ — количество значений с $cnt \equiv 0 \pmod 4$ (то есть кратно 4)

Теоретически максимально возможный ответ — это $x + 2y + 2z$, поскольку:

- Значения с нечётным $cnt$ всегда дают вклад 1 (ровно одно из $u$ или $v$ будет нечётным)
- Значения с $cnt \equiv 2 \pmod 4$ могут дать вклад 2, если взять $u = v = \frac{cnt}{2}$
- Значения с $cnt \equiv 0 \pmod 4$ также могут дать вклад 2, если выбрать, например, $u = \frac{cnt}{2} + 1$, $v = \frac{cnt}{2} - 1$

Для каждого числа при распределении между $p$ и $q$ возникает дисбаланс $d = u - v$. Если $|p| = |q| = n$, то сумма всех таких дисбалансов должна быть равна 0. Рассмотрим, какие дисбалансы возникают при оптимальном распределении (с вкладом 2):

- Для числа с $cnt \equiv 2 \pmod 4$ можно выбрать $u = v = \frac{cnt}{2}$, тогда дисбаланс $d = 0$.
- Для числа с $cnt \equiv 0 \pmod 4$ можно выбрать $u = \frac{cnt}{2} + 1$, $v = \frac{cnt}{2} - 1$, тогда дисбаланс $d = 2$ (или $d = -2$, если поменять местами).

Стоит отметить, что количество чисел с нечётным $cnt$ ($x$) всегда чётно. Доказательство: сумма всех $cnt$ равна $2n$ (чётное число). Сумма чётных частот чётна. Поэтому сумма нечётных частот должна быть чётной, что возможно только если количество нечётных частот чётно.

**Случай 1: $z$ чётно, $x$ - произвольное**

Для чисел с нечётным $cnt$ мы можем распределить их дисбалансы как $-1, +1, -1, +1, \ldots$. Поскольку $x$ чётно, сумма таких чередующихся дисбалансов будет 0. Если $x = 0$, то сумма дисбалансов очевидно будет 0.

Если $z$ чётно, то мы можем разбить числа с $cnt \equiv 0 \pmod 4$ на пары. В каждой паре одно число распределим с $d = +2$, другое с $d = -2$, чтобы дисбалансы $d_i$ чередовались: $+2, -2, +2, -2, \ldots$. Тогда их вклады компенсируются, и суммарный дисбаланс от всех таких чисел будет 0.

Для чисел с $cnt \equiv 2 \pmod 4$ ($y$ штук) ситуация проще. Поскольку $cnt$ даёт остаток 2 при делении на 4, мы можем выбрать симметричное распределение: $u = v = \frac{cnt}{2}$. Такое распределение всегда даёт дисбаланс $d = u - v = 0$. Суммарный дисбаланс 0.

Итого получим суммарный дизбаланс всех чисел 0, а значит условие $|p| = |q|$ выполнится.

**Случай 2: $z$ - нечётно, $x > 0$**

Для чисел с нечётным $cnt$ мы можем распределить их дисбалансы как $-1, +1, -1, +1, \ldots$. Поскольку $x$ чётно, сумма таких чередующихся дисбалансов будет 0. После чего изменим последнюю $+1$ на $-1$ и получим суммарный дисбаланс $-2$.

Для чисел с $cnt \equiv 0 \pmod 4$ мы можем распределить их так, чтобы дисбалансы $d_i$ чередовались: $+2, -2, +2, -2, \ldots$. Если $z$ нечётно, то в такой последовательности будет на один $+2$ больше, чем $-2$, и суммарный дисбаланс будет $D = +2$.

Для чисел с $cnt \equiv 2 \pmod 4$ ($y$ штук) ситуация проще. Поскольку $cnt$ даёт остаток 2 при делении на 4, мы можем выбрать симметричное распределение: $u = v = \frac{cnt}{2}$. Такое распределение всегда даёт дисбаланс $d = u - v = 0$. Суммарный дисбаланс 0.

Итого получим суммарный дизбаланс всех чисел 0, а значит условие $|p| = |q|$ выполнится.

**Случай 3: $z$ - нечётно, $x = 0$**

Для чисел с $cnt \equiv 0 \pmod 4$ мы можем распределить их так, чтобы дисбалансы $d_i$ чередовались: $+2, -2, +2, -2, \ldots$. Если $z$ нечётно, то в такой последовательности будет на один $+2$ больше, чем $-2$, и суммарный дисбаланс будет $D = +2$.

Для чисел с $cnt \equiv 2 \pmod 4$ ($y$ штук) ситуация проще. Поскольку $cnt$ даёт остаток 2 при делении на 4, мы можем выбрать симметричное распределение: $u = v = \frac{cnt}{2}$. Такое распределение всегда даёт дисбаланс $d = u - v = 0$. Суммарный дисбаланс 0.

Теперь мы имеем суммарный дисбаланс $+2$ от чисел с $cnt \equiv 0 \pmod 4$ и 0 от остальных. Общая сумма дисбалансов равна $+2$, что означает, что $|p| - |q| = 2$, то есть группы имеют разный размер. Это противоречит условию задачи, так как обе группы должны содержать ровно $n$ элементов.

Мы не можем использовать числа с нечётной частотой для компенсации этого дисбаланса, поскольку в этом случае $x = 0$ (нет чисел с нечётным $cnt$). Единственный способ исправить ситуацию — изменить распределение одного из чисел с $cnt \equiv 0 \pmod 4$ так, чтобы оно давало вклад 0 вместо 2.

Для одного числа с $cnt \equiv 0 \pmod 4$ вместо распределения $u = \frac{cnt}{2} + 1$, $v = \frac{cnt}{2} - 1$ (которое даёт вклад 2 и дисбаланс $+2$) выберем симметричное распределение $u = v = \frac{cnt}{2}$. Это даст:

- Дисбаланс $d = 0$ (вместо $+2$). Суммарный дисбаланс станет $+2 - 2 = 0$.

- Вклад 0 (вместо 2), так как $u$ и $v$ теперь оба чётны. Суммарный ответ уменьшился на 2 (из-за снижения вклада).

Теперь количество чисел с $cnt \equiv 0 \pmod 4$, дающих вклад 2, стало $z - 1$ (чётное число, так как $z$ было нечётно). Оставшиеся $z - 1$ таких чисел можно распределить попарно с дисбалансами $+2$ и $-2$, чтобы сумма их дисбалансов была 0.

Максимально достижимый ответ в этом случае равен $2y + 2z - 2$

# Максимизация суммы побитовых И - легкая версия

Для заданного $n$ рассмотрим все перестановки $p$ массива $[0, 1, \ldots, 2^n - 1]$. Нужно найти лексикографически минимальную перестановку, максимизирующую сумму:

$$S(p) = \sum_{i=0}^{2^n-1} \text{popcount}(p_0 \& p_1 \& \ldots \& p_i)$$

где popcount$(x)$ — количество единичных битов в двоичном представлении $x$, а & — побитовое И.

Здесь $n \leq 3$.

---

Заметим, что операция побитового И обладает свойством монотонности: при последовательном применении И к числам результат может только уменьшать количество единичных битов. Конкретно:

$$\text{popcount}(a \& b) \leq \min(\text{popcount}(a), \text{popcount}(b))$$

Также важно, что И чисел сохраняет только те биты, которые установлены в 1 во всех операндах. Поэтому для максимизации суммы $S(p)$ нужно начинать с чисел, имеющих максимальное количество единичных битов и располагать числа так, чтобы при последовательном применении И как можно дольше сохранялось большое количество единичных битов

**Для $n = 1$:** Числа: 0, 1

- Перестановка [1, 0]: $S = \text{popcount}(1) + \text{popcount}(1\&0) = 1 + 0 = 1$
- Перестановка [0, 1]: $S = \text{popcount}(0) + \text{popcount}(0\&1) = 0 + 0 = 0$

**Для $n = 2$:** Числа: 0, 1, 2, 3 (в двоичном виде: 00, 01, 10, 11)

- Начинаем с 3 (11) — наибольшее количество единичных битов (2)
- Далее выбираем числа, сохраняющие как можно больше единичных битов
- После вычисления всех вариантов оптимальная перестановка: [3, 1, 0, 2]

**Для $n = 3$:** Числа: 0, 1, 2, 3, 4, 5, 6, 7 (в двоичном виде: 000, 001, 010, 011, 100, 101, 110, 111)

- Начинаем с 7 (111) — наибольшее количество единичных битов (3)
- Далее выбираем числа, сохраняющие как можно больше единичных битов
- После вычисления всех вариантов оптимальная перестановка: [7, 3, 1, 5, 0, 2, 4, 6]

# Максимизация суммы побитовых И - сложная версия

Для заданного $n$ рассмотрим все перестановки $p$ массива $[0, 1, \ldots, 2^n - 1]$. Нужно найти лексикографически минимальную перестановку, максимизирующую сумму:

$$S(p) = \sum_{i=0}^{2^n-1} \text{popcount}(p_0 \& p_1 \& \ldots \& p_i)$$

где popcount$(x)$ — количество единичных битов в двоичном представлении $x$, а & — побитовое И.

Здесь $n \leq 16$.

---

Рассмотрим префикс перестановки длины $i + 1$. Пусть $m_i = p_0 \& p_1 \& \ldots \& p_i$ — результат побитового И всех чисел в префиксе. Тогда popcount$(m_i)$ равно количеству битов, которые установлены в 1 во всех числах этого префикса.

Для того чтобы popcount$(m_i) \geq k$, необходимо, чтобы существовало $k$ битовых позиций, в которых все числа префикса имеют 1. Так как каждое число — это $n$-битное число от 0 до $2^n - 1$, то количество чисел, имеющих фиксированные $k$ единичных битов, равно $2^{n-k}$.

Таким образом, максимальное количество префиксов с popcount $\geq k$ не может превышать $2^{n-k}$, так как после выбора чисел с фиксированными $k$ единичными битами у нас остаётся только $n-k$ свободных битов, которые могут принимать $2^{n-k}$ различных значений.

Мы хотим достичь этой верхней границы для всех $k$. Один из способов — расположить числа в порядке убывания: $2^n-1, 2^n-2, \ldots, 0$. Однако нам нужно лексикографически минимальное решение.

Будем строить перестановку итеративно: начнём с числа $2^n-1$ (все биты равны 1). Далее будем добавлять числа, которые отличаются от текущего результата И только в одном бите, сбрасывая очередной бит в 0. После сброса всех битов начинаем добавлять оставшиеся числа в лексикографическом порядке.

- Другое решение - рассмотрим следующий алгоритм: для $i$ от 0 до $2^n-1$ положим

$$p_i = \text{reverse\_bits}(i) \oplus (\text{reverse\_bits}(i) \gg 1)$$

  где reverse_bits — обращение порядка битов (под "обращением порядка битов"(bit reversal) числа $x$ длины $n$ бит понимаем число $\text{rev}(x)$, у которого биты расположены в обратном порядке), $\oplus$ — XOR, $\gg$ — сдвиг вправо. Это соответствует известной последовательности Gray code (код Грея), но в обратном порядке.

## Саша и казино

Саша приходит в казино с $a$ монетами. Правила:

- Нельзя проиграть более $x$ раз подряд
- При ставке $y$ монет ($0 < y \leq h$, где $h$ — текущий капитал):
  - В случае победы получает $y \cdot k$ монет (чистый выигрыш: $y \cdot (k-1)$)
  - В случае поражения теряет $y$ монет

Может ли Саша выбрать стратегию ставок так, чтобы гарантированно увеличить свой капитал до сколь угодно большой величины?

---

Давайте заметим: условие, что мы можем получить сколь угодно большое значение означает, что нам необходимо получить гарантированно хотя бы +1 к нашим монетам. В этом случае мы можем повторять данную стратегию бесконечно много раз.

Также заметим, что если мы до этого проиграли суммарно $z$, то в следующий раунд нам необходимо поставить $y$ такое, чтобы $y \cdot (k-1) > z$, так как иначе казино может отдать нам победу. В этом случае условие на проигрыш не более $x$ раз подряд пропадет, а мы ушли в минус. А следовательно тактика является не оптимальной.

А значит решение — мы ставим сначала 1, затем мы ставим минимальное число, чтобы победа перекрыла наш проигрыш. И если нам хватит на $x+1$ такую ставку, то казино обязано уйти в минус, иначе же мы не сможем победить.

И получается решение за $O(x)$, где мы просто циклом считаем эти значения.

## Ян и доска

На доске написаны $n$ различных целых чисел $x_1, x_2, \ldots, x_n$. Можно выполнять операцию: выбрать два числа $x, y$ (не обязательно различных) с доски и записать на доску число $2x-y$ (исходные числа остаются на доске). Может ли после нескольких таких операций на доске появиться число $k$?

---

Рассмотрим операцию $2x-y$. Заметим, что если мы рассматриваем разности между числами, то эта операция сохраняет разность между числами по модулю некоторого значения.

Пусть $d = x_2 - x_1$ — разность между двумя числами. Тогда операция $2x-y$ может быть переписана как:

$$2x - y = x + (x - y)$$

То есть мы добавляем к числу $x$ разность между $x$ и $y$.

Пусть $g = \gcd(|x_2 - x_1|, |x_3 - x_1|, \ldots, |x_n - x_1|)$. Тогда заметим, что любое число, которое можно получить, имеет вид:

$$x_i + m \cdot g$$

для некоторого целого $m$. Докажем этот факт:

- Исходные числа имеют эту форму
- Если $a = x_i + m_1 \cdot g$ и $b = x_j + m_2 \cdot g$, то:

  $$2a - b = 2(x_i + m_1 \cdot g) - (x_j + m_2 \cdot g) = (2x_i - x_j) + (2m_1 - m_2) \cdot g$$

  что также имеет нужный вид

Таким образом, число $k$ может быть получено тогда и только тогда, когда:

$$k \equiv x_i \pmod{g}$$

для некоторого $i$, то есть разность $k - x_i$ должна делиться на $g$ для некоторого $i$.

## Удаление последовательности - легкая версия

У Поликарпа есть последовательность всех натуральных чисел от 1 до $10^{12}$. Он $x$ раз выполняет операцию: одновременно удалить все числа на позициях $y, 2 \cdot y, 3 \cdot y, \ldots, m \cdot y \leq n$, где $n$ — текущая длина последовательности. После этого нужно найти $k$-е число в оставшейся последовательности или определить, что длина результата меньше $k$. Здесь $x \leq 10^5$

---

Во многих задачах, где необходимо искать $k$-й элемент, появляется бинарный поиск. Давайте и в этой задаче попытаемся применить бинарный поиск по ответу. Осталось только понять, как будет выглядеть функция проверки.

Пусть мы зафиксировали число $p$ и нам бы хотелось понять, сколько чисел останется в итоге, если изначальная последовательность состояла из всех натуральных чисел от 1 до $p$. Так как в каждом действии Поликарп удаляет все элементы последовательности, чей номер кратен $y$, то количество удаленных элементов после одного действия можно посчитать по формуле $\lfloor \frac{p}{y} \rfloor$. Ограничения в простой версии задачи позволяют нам вычислять длину оставшейся последовательности после каждого действия Поликарпа, поэтому длину результирующей последовательности можно считать так:

```
for (int i = 0; i < x; i++) {
    p -= p / y;
}
```

Если мы найдем такое минимальное $p$, что после всех действий Поликарпа останется ровно $k$ элементов, то такое $p$ и будет являться ответом.

Несложно показать, что наша функция монотонна, поэтому мы справедливо можем использовать бинарный поиск.

## Удаление последовательности - сложная версия

У Поликарпа есть последовательность всех натуральных чисел от 1 до $10^{12}$. Он $x$ раз выполняет операцию: одновременно удалить все числа на позициях $y, 2 \cdot y, 3 \cdot y, \ldots, m \cdot y \leq n$, где $n$ — текущая длина последовательности. После этого нужно найти $k$-е число в оставшейся последовательности или определить, что длина результата меньше $k$. Здесь $x \leq 10^{12}$

---

В отличие от простой версии мы не можем вычислять длину последовательности после каждого действия Поликарпа, так как $x$ может быть довольно большим. Однако заметим следующее: мы каждый раз вычитаем число $\frac{p}{y}$, и при этом $y$ фиксирован, из этого следует, что различных значений вычитаемых чисел не более чем $\sqrt{p}$, поэтому мы можем объединить действия Поликарпа и вычитать их сразу группами:

```
for (int i = 0; i < x;) {
    int cur_value = p / y;
    if (cur_value == 0) {
        break;
    }
    int where_cur_value_changes = cur_value * y - 1;
    int actions_in_group = (p - where_cur_value_changes + cur_value - 1) / cur_value;
    actions_in_group = min(x - i, actions_in_group);
    p -= actions_in_group * cur_value;
    i += actions_in_group;
}
```

Такая оптимизация позволит нам написать решение за $O(\sqrt{A} \cdot \log A)$. Однако такое решение все еще медленное для того, чтобы получить Accepted.

Оказывается, мы можем избавиться от бинарного поиска по ответу. Для этого давайте рассмотрим обратную задачу.

Ранее мы фиксировали $p$ и вычитали удаленные элементы, то есть после одной операции:

$$p' = p - \left\lfloor \frac{p}{y} \right\rfloor$$

Но мы можем попытаться по $p'$ найти $p$, оказывается, мы можем сделать это следующим образом: среди каждых $y$ подряд идущих позиций одна кратная $y$ удаляется, а $y-1$ остаются. Значит, чтобы получить $p'$-й оставшийся элемент, мы берем: $p'$ самих оставшихся элементов; плюс по одному удалённому на каждый полный блок из $y-1$ оставшихся. Количество таких полных блоков — $\left\lfloor \frac{p'-1}{y-1} \right\rfloor$.

$$p = p' + \left\lfloor \frac{p'-1}{y-1} \right\rfloor$$

Таким образом, если мы начнем с позиции $k$ и применим $x$ таких обратных операций, то мы найдем ответ. Асимптотическая оценка времени: $O(x)$, но с оптимизацией группировки операций становится $O(\sqrt{A})$, где $A$ зависит от длины изначальной последовательности, в данной задаче $A = 10^{12}$.