# Задача A. Монетки

*Автор и разработчик задачи: Даниил Голов*

Нам дана последовательность из $n$ цифр 0 или 1. Затем, продолжается следующий процесс:

- Если последовательность не содержит 1, процесс останавливается.
- Иначе, пусть $k$ — количество 1. Инвертируется элемент на позиции $k$ (в 1-индексации).

Нужно либо сказать, за какое количество операций процесс остановится, либо, что он будет продолжаться бесконечно.

Рассмотрим инвертирование элемента на позиции $k$. Заметим, что до инвертирования существовал элемент на позиции $\geqslant k$, равный 1. Пусть он находился на позиции $q$. Тогда сначала $q-k$ раз мы заменим 0 на 1 на позициях $[k, q)$, затем элемент на позиции $q$ поменяется с 1 на 0, и затем еще $q-k$ раз мы заменим 1 на 0 на позициях $[k, q)$. Таким образом, мы сделаем $(q-k)\cdot 2+1$ операций, после чего количество единиц уменьшится на 1. Отсюда следует, что процесс всегда конечный. А также, отсюда понятен алгоритм решения задачи за время $O(n \cdot \log(n))$. Нужно поддерживать множество позиций 1, и за один раз обрабатывать удаление одной единицы.

# Задача B. Электронный замок

*Автор и разработчик задачи: Арсений Кириллов*

Заметим, что мы всегда хотим получить более длинное число, так как длинное число больше короткого. Тогда для каждого $n$ можно посчитать метод динамического программирования, какую максимальную длину можно получить, если включено ровно $n$ сегментов. $len[n] = 1 + \max\limits_{x \in D}(len[n - cost[x]])$, где $cost[x]$ — количество горящих сегментов у цифры $x$, а $D$ — множество доступных цифр.

Для вывода самого пароля, достаточно каждый раз выводить максимальную цифру, которая уменьшает возможную длину ровно на 1, а также не забыть, что первая цифра не должна быть нулевой.

# Задача C. Лемурьи вечеринки

*Автор задачи и разработчик: Даниил Орешников*

Для решения этой задачи достаточно переформулировать ее в терминах сочетаний. Если $C_a^b$ — число способов выбрать $b$ элементов из $a$ различных, то ответом на задачу будет

$$\sum_{t=0}^{k} C_k^t \cdot C_{k-t}^{n-2t}$$

Действительно, нам надо выбрать на $n$ мест различными способоми какие виды лемуров будут представлены двумя лемурами, а какие — одним. Переберем количество «пар» лемуров $t$. Для фиксированного $t$: выбрать $t$ пар можно $C_k^t$ способами, а на оставшиеся $n-2t$ места надо разместить по одному лемуру из некоторых из оставшихся $k-t$ видов.

Теперь вспомним, что $C_n^k = \frac{n!}{k!(n-k)!}$ (где $x!$ — произведение всех чисел от 1 до $x$. Пользуясь модульной арифметикой, предподсчитаем $\texttt{frac}[\texttt{i}] = i! \bmod M$ (где за $M$ обозначен простой модуль $10^9+7$) как $\texttt{frac}[i+1] = \texttt{frac}[i] \cdot (i+1) \bmod M$.

Чтобы реализовать деление, воспользуемся расширенным алгоритмом Евклида для нахождения обратного по простому модулю и запомним все обратные к факториалам, после чего $C_n^k$ можно считать за $\mathcal{O}(1)$, используя предподсчитанные значения факториалов и обратных к ним величин по модулю $M$. Суммарное время работы — $\mathcal{O}(n \log n)$ на предподсчет (при этом log — достаточно завышенная оценка) и $\mathcal{O}(n)$ на вычисление ответа.

# Задача D. Дом в невысоком дереве

*Автор задачи: Орешников Даниил, разработчик задачи: Николай Будин*

Оптимальное расположение лестниц строится конструктивно. Во-первых нужно поставить $n$ лестниц между центральными комнатами на каждом этаже. Затем, пока количество оставшихся

лестниц хотя бы $n$, нужно соединять лестницами все комнаты на одной вертикали. И если осталось меньше $n$ лестниц, их тоже нужно поставить на одну вертикаль подряд.

После чего, нужно посчитать ответ. Ограничения позволяют сделать это за $O(n^2)$, запустив от каждой из комнат bfs.

# Задача E. Подсчет операций

*Автор задачи: Даниил Орешников, разработчик: Ильдар Загретдинов*

Заметим, что для того, чтобы обнулить значение индикатора в листе дерева, нам необходимо сделать модуль от веса листа операций, при этом все вершины на пути от листа до корня тоже невозможно не затронуть. Поэтому минимальное количество операций для обнуления веса всех листьев — сумма модулей весов листьев в начале. После этого ненулевыми (возможно) остались веса вершин на предпоследнем уровне, для их обнуления придется сделать аналогичные операции. Заметим, что выполняя поиск в глубину, мы понимаем на каждом шаге, сколько нам нужно сделать операций для обнуления значения в текущей вершине (значение в вершине после применений операций к детям изменилось на сумму весов детей). Таким образом мы понимаем, сколько нужно сделать операций применительно к этой вершине.

# Задача F. Черные и белые

*Автор идеи: Мухаммаджон Хакимов, разработчик задачи: Григорий Хлытин*

Если будем считать формулу итеративно, получим вердикт TL, т.к. такое решение будет работать за время $\mathcal{O}(n)$, где $n$ — количество ходов в игре ($1 \leqslant n \leqslant 10^{18}$).

Заметим, что если расписать формулу искомой вероятности как $p = \left(1 - \frac{1}{2^2}\right) \cdot \left(1 - \frac{1}{3^2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{(n+1)^2}\right)$, что равно $\left(\frac{2^2-1}{2^2}\right) \cdot \left(\frac{3^2-1}{3^2}\right) \cdot \ldots \cdot \left(\frac{(n+1)^2-1}{(n+1)^2}\right)$. Если раскрыть числители как разность квадратов, получим

$$p = \left(\frac{(2-1)(2+1)}{2^2}\right) \cdot \left(\frac{(3-1)(3+1)}{3^2}\right) \cdot \ldots \cdot \left(\frac{((n+1)-1)((n+1)+1)}{(n+1)^2}\right)$$

Сокращаем знаменатели всех дробей, кроме крайних, с числителями соседних и получаем ответ

$$p = \frac{n+2}{2(n+1)}$$

Так как числа $n+2$ и $n+1$ всегда взаимно просты, то чтобы получить несократимую дробь, остается только рассмотреть случай $(n+2) \bmod 2 = 0$, когда надо разделить числитель и знаменатель на 2. Суммарное время работы — $\mathcal{O}(1)$.