# Problem A. 1024 Stack Edition

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Для начала рассмотрим случай, когда $N = 0$. Пусть $f(i)$ — это ожидаемое наименьшее количество монет, которое потребуется потратить, чтобы собрать число $2^i$. Приведем формулы для подсчета $f(i)$ в случае, если $p < 100$. Если же $p = 100$, то эти же формулы адаптируются тривиальным образом.

1. $f(0) = 1.0/p - 1$;
2. $f(1) = min(f(0) \cdot 2,\ 1/(1-p) - 1,\ 1 - p)$;
3. $f(i) = 2 \cdot f(i-1),\ i > 1$.

В случае с $N = 0$ ответом на задачу будет $f(10)$.

Теперь предположим, что $N > 0$. Обозначим через $g(i, j)$ ожидаемое наименьшее количество монет, которые нам нужно будет потратить, чтобы из последних (верхних) $i$ элементов стека получить только одно число $2^j$. Подсчет $g(i, *)$ для фиксированного i будет происходить в 2 этапа (через $S(i)$ обозначим двоичный логарифм $i$-го сверху элемента стека):

1. Посчитаем $g(i, S(i)) = 1 + min(g(i-1, *)),\ i > 1$. Этот случай соответствует тому, что из всего, что выше элемента $i$, мы получаем некоторое число, после чего удаляем его.

   Также посчитаем $g(i, S(i)+1) = g(i-1, S(i))$. Этот случай соответствует тому, что мы из всех элементов, расположенных выше $i$, собираем число $2^{S(i)}$, после чего объединяем его с текущим элементом, который также равен $2^{S(i)}$. Для всех $j$, отличных от $S(i)$ и $S(i)+1$, положим $g(i, j) = +\inf$.

2. На первом этапе мы "разобрались" с числом $S(i)$. Прежде чем переходить к $g(i+1, *)$ и числу $S(i+1)$, мы можем произвести некоторые действия с $i$-м элементом стека, превратив его в любой другой элемент. Чтобы это учесть в динамике, пересчитаем $g(i, *)$ следующим образом:

   - $g(i, 0) = min(g(i, 0), min(g(i, *)) + 1.0/p)$ — мы либо используем уже найденное значение $g(i, 0)$, либо берем любое число, удаляем его и ставим туда $1 = 2^0$.
   - $g(i, j) = min(g(i, j), min(g(i, *)) + 1 + f(j), g(i, j-1) + f(j-1)),\ j > 0$. Здесь мы либо удаляем стоявшее на месте $i$ число и затем за $f(j)$ монет ставим туда число $2^j$, либо ставим за $g(i, j-1)$ шагов число $2^{j-1}$, собираем за $f(j-1)$ шагов еще $2^{j-1}$ и объединяем их.

Ответом на задачу будет $g(N, 10)$. Следует обратить внимание, что в промежуточных вычислениях следует также считать $g(*, 11)$. Сложность решения $O(NR)$, где $R$ — максимальная степень двойки, которая присутствует во входных данные (в данном случае $R \leq 10$).

# Problem B. Строчная задача

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 4 seconds |
| Memory limit: | 512 mebibytes |

Рассмотрим пару строк $s$ и $w$. Строка $s$ почти меньше строки $w$ тогда и только тогда, когда минимальная лексикографически строка $s^{min}$, которую можно получить из $s$ заменой одной буквы, строго меньше $w$.

Однако, как легко видеть, $s^{min}$ получается из строки $s$ заменой первого символа не равного '`a`' на символ '`a`'. Если строка $s$ состоит только из символов '`a`', то $s = s^{min}$.

Рассмотрим строку $s_i$, и найдем $C_i$ — количество строк $s_j$ (возможно $i = j$) таких, что $s_i^{min} < s_j$. Тогда к ответу нужно прибавить $C_i - 1$, если $s_i^{min} \neq s_i$, и $C_i$ в противном случае.

Для нахождения величин $C_i$ можно воспользоваться двумя способами:

- отсортировать строки $s_i$ и находить $C_i$ двоичным поиском;
- добавить все строки в `trie`, и находить $C_i$ спуском по нему.

# Problem C. Кубик

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Основное, что надо заметить в этой задаче — что гексамино можно сгибать в разные стороны. При этом получаются два кубика, отличающихся друг от друга только взаимной перестановкой одной пары противоположных цифр (так как поворот на 180 градусов переставляет две пары противоположных цифр, то перестановка всех трёх пар эквивалентна перестановке одной пары).

Далее существует несколько методов решения.

Один из них состоит из следующих шагов:

1. Устанавливаем кубик нижней стороной на соответствующую клетку гексамино
2. Для каждого из четырёх поворотов кубика вокруг своей оси начинаем перекатывать его по гексамино до тех пор, пока возможно перекатить кубик так, чтобы нижняя грань попала на клетку с соответствующим номером.
3. Если для какого-то из поворотов все 6 клеток посещены, то ответ "`Yes`".
4. Если нет, то меняем местами цифры на правой и левой гранях и повторяем процедуру с шага 2. Если и в этом случае ни при одном из поворотов не посещены все 6 клеток, то ответ — "`No`".

Также существует возможность предпросчитать все 11 развёрток и затем проверять совпадение гексамино (поворачивая его в стандартное положение), а в случае совпадения — соответствие пар чисел, использованных для нумерации противоположных сторон кубика, с теми же парами в развёртке.

# Problem D. НЛО

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Это самая простая задача набора. Очевидно, что размерность параллелепипеда не меньше количества различных длин рёбер. Также заметим, что в $K$-мерном параллелепипеде $2^{K-1}$ рёбер одного типа, то есть в случае наличия $X$ одинаковых рёбер для них будет использовано $[(X-1)/2^{K-1}]+1$ измерение.

Поэтому в порядке возрастания перебираем все размерности, пока вычисленное по указанной выше формуле суммарное количество использованных измерений не будет меньше или равно текущей размерности. Так как при $K=21$ рёбер одного типа будет $2^{20} > 10^6$, то перебор будет небольшим.

# Problem E. Недополняемое паросочетание

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Для решения задачи заметим, что число вершин в левой доле невелико.

Будем считать следующую динамику, добавляя по одной вершине из правой доли: $f(i, j)$ — количество способов расставить рёбра в случае, когда мы рассмотрели первые $i$ вершин правой доли, а вершины левой доли заданы троичной маской $j$ следующим образом: 0 — вершина не могла быть соединена ребром ни с одной вершиной из правой доли, 1 — вершина могла быть соединена ребром ни с одной вершиной из правой доли, однако мы не поставили это ребро, 2 — вершина соединена ребром с одной из вершин правой доли.

Таким образом, рассматривая очередную вершину из правой доли, мы можем пересчитать маску для вершин левой доли. Очевидно, что при подсчете ответа нужно рассматривать маски без 1. Таким образом, ответом будет сумма $f(n_2, ValidMask)$, для всех троичные масок $ValidMask$, не содержащих единиц в троичной записи.

# Problem F. Музыкальный мир

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Обозначим как $Y_i$ случайную величину, равную количеству песен в $i$-м поколении.

Предлагаемое решение задачи основано на понятии производящей функции. Если вы не знакомы с ним, ознакомиться можно, к примеру, в статье Википедии

`http://ru.wikipedia.org/wiki/Производящая_функция_последовательности`

Рассмотрим производящие функции заданных во входе случайных величин $\psi_i(x) = \sum\limits_{j=0}^{K_i} p_{ij}x^j$, а также производящие функции случайных величин $Y_i$ $\varphi_i(x) = \sum\limits_{j=0}^{\infty} \mathbb{P}\{Y_i = j\}x^j$.

У нас есть явный вид всех $\psi_i$, а также явный вид $\varphi_1(x) = x$. Научимся выражать $\varphi_{i+1}$ через $\psi_i$ и $\varphi_i$.

В случае, если бы мы точно знали, что $Y_i = k$, то можно было бы утверждать, что $\varphi_{i+1}(x) = \psi_i(x)^k$. Но, так как $Y_i$ случайная величина, то искомая производящая функция есть сумма производящих функций, соответствующих разным значениям, с весами, равными вероятностям этих значений, то есть

$$\sum_{j=0}^{\infty} \mathbb{P}\{Y_i = j\}\psi_i(x)^j = \varphi_i(\psi_i(x)).$$

Так как максимальное значение искомой величины очень большое, вычислить явный вид производящей функции для $Y_{n+1}$ не представляется возможным. Однако он нам и не нужен. Несложно заметить, что величина, которую необходимо посчитать

$$\mathbb{E}\frac{Y_i(Y_i-1)}{2} = \sum_{j=0}^{\infty} \mathbb{P}\{Y_i = j\}\frac{j(j-1)}{2} = \frac{1}{2}\varphi_i''(1).$$

Научимся пересчитывать значение, первую и вторую производную функции $\varphi_{i+1}(x)$ в (1) через предыдущие функции. Напомню, что $\varphi_i(1) = \psi_i(1) = 1$. Как мы знаем, $\varphi_{i+1}(x) = \varphi_i(\psi_i(x))$. Тогда

$$\varphi'_{i+1}(x) = \varphi'_i(\psi_i(x))\psi'_i(x) \text{ и}$$

$$\varphi''_{i+1}(x) = \varphi''_i(\psi_i(x))\psi'_i(x)^2 + \varphi'_i(\psi_i(x))\psi''_i(x).$$

При подстановке $x = 1$ эти формулы упрощаются и принимают вид

$$\varphi'_{i+1}(1) = \varphi'_i(1)\psi'_i(1)\text{и}$$

$$\varphi''_{i+1}(1) = \varphi''_i(1)\psi'_i(1)^2 + \varphi'_i(1)\psi''_i(1).$$

Также несложно понять, что вычисления можно проводить сразу по модулю $10^9+7$, заменив исходные вероятности на соответствующие значения по модулю.