# Задача A. Рейтинг на codeforces

Решим задачу жадным алгоритмом. Рассмотрим по-очереди элементы массива от 1-го до $n$-го. Каждый раз будем из двух вариантов выбирать тот, который во-первых, больше либо равен предыдущего выбранного значения, а во-вторых, если все равно оба варианта возможны, то меньший их двух (то есть, отрицательный). Если на каждом шаге получилось выбрать очередной элемент, то решение найдено. Иначе, решения не существует.

# Задача B. Чип, Дейл и прямоугольник

Для начала заметим что наибольшую площадь прямоугольник имеет тогда, когда он квадрат, тогда если площадь этого квадрата меньше $s$, то ответа не существует, иначе он всегда есть. Пусть $A(x_1, y_1)$, $B(x_2, y_2)$, $O(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$ – то есть середина диагонали. Тогда, если вектор $\vec{AO}$ повернуть на угол между диагоналями искомого прямоугольника, а затем отложить от точки $O$ в обе стороны, мы найдем искомые точки. Найти угол можно из формулы $S = \frac{d^2 * sin(alpha)}{2}$, где $S$ – площадь прямоугольника, $d$ – длина его диагонали, а $alpha$ – угол между диагоналями. Тогда $alpha = arcsin(\frac{2*S}{d_1 * d_2})$. Далее можно используя формулу поворота вектора на заданный угол найти нужный вектор.

# Задача C. Болик, Лёлик и странное поручение

Будем решать задачу по высоте и ширине отдельно. Очевидно, что чтобы покрыть отрезок длины $h$ отрезками длины $a$ понадобится минимум $\lceil \frac{h}{a} \rceil$ отрезков. Аналогично для ширины. Осталось только перемножить.

Таким образом ответ на задачу: $\lceil \frac{h}{a} \rceil \times \lceil \frac{w}{b} \rceil$

# Задача D. Петя и Вася снова играют в игру

Если $s = 1$, то количество ходов, которое сделают игроки фиксировано и равно $n \cdot m$. Тогда Петя выиграет только если это число нечётное.

Теперь считаем, что $s > 1$. Если одна из сторон нечётная или $s \geqslant 4$, то Петя может первым ходом построить прямоугольник, симметричный относительно центра поля. После этого он может ходить центрально-симметрично ходам Васи. Значит, если Вася сделал ход, его сможет сделать и Петя. Значит, Петя выиграет.

Если же ни одно из этих условий не выполняется, то есть обе стороны чётны и $2 \leqslant s \leqslant 3$, то никакой игрок своим ходом не может построить прямоугольник, содержащий две центрально-симметричные клетки. Поэтому Вася может делать ходы центрально-симметричные ходам Пети. Значит, Петя проиграет.

# Задача E. Солдатики

Эту задачу можно решать бинпоиском по ответу. Для этого достаточно заметить, что если все солдатики могут собраться за $t$ действий, то они могут собраться и за количество действий, большее $t$, просто сделав несколько одинаковых шагов после, а поэтому бинпоиск можно применять.

Давайте теперь проверим, смогут ли солдатики собраться за $t$ действий. Каждый солдатик может оказаться в любой точке, для которой *манхэтеннское расстояние* меньше или равно $t$ (манхэттенским расстоянием называется сумма расстояний по двум координатам).

Несложно заметить, что множество точек, удаленных от данной по манхэттенскому расстоянию не более, чем на заданную константу — это правильный ромб. Таким образом, необходимо пересечь ромбы, задающие достижимые для солдатиков за $t$ действий точки, и проверить, содержит ли оно хотя бы одну точку с целочисленными координатами. Воспользуемся матрицей поворота, чтобы повернуть систему и, для простоты, искать пересечения квадратов (`https://ru.wikipedia.org/wiki/Матрица_поворота`).

Время работы такого решения — $\mathcal{O}(n \log C)$, где $C$ — максимальная начальная координата содатика.

# Задача F. Обезвредить бомбу

Рассмотрим число $a + b + x$ и заметим, что оно обязано делиться как на $a$, так и на $b$, так как

- $a + x \vdots b$, $b \vdots b$, значит $a + x + b \vdots b$

- $b + x \vdots a$, $a \vdots a$, значит $b + x + a \vdots a$

Делимость на $a$ и на $b$ равносильна делимости на $\texttt{lcm}(a, b)$, где `lcm` обозначает *наименьшее общее кратное*, которое можно найти как $\frac{ab}{\texttt{gcd}(a,b)}$, а `gcd` или *наибольший общий делитель* можно найти с помощью алгоритма Евклида.

Итак, обозначим наименьшее общее кратное чисел $a$ и $b$ за $y$, тогда задача сводится к поиску наименьшего неотрицательного $x$, что $x+a+b$ делится на $y$. Очевидно, что тогда любой подходящий нам $x$ имеет вид $ky - a - b$ для некоторого целого $k$.

Осталось выбрать среди таких минимальный. Несложно показать, что $a, b \leqslant y$, а значит $x_2 = 2y-a-b \geqslant 0$, соответственно, если ответ меньше, чем $x_2$, то это может быть только $x_1 = y-a-b$, так как дальше идут только отрицательные числа ($-a - b < 0$). Получаем следующий ответ — если $y \geqslant a + b$, то это $y - a - b$, иначе $2y - a - b$.

# Задача G. Чебурашка и непонятный прибор

Если $a \leqslant b$, то надо просто $b - a$ раз нажать на кнопку и получить нужное число.

Если $a > b$, то ситуация значительно усложняется: надо сначала $m - a$ раз нажать на кнопку и получить число $m$, затем нажать один раз и получить число 1, а потом нажать $b - 1$ раз и получить $b$. Итого будет произведено $(m - a) + 1 + (b - 1) = m + b - a$ нажатий.

# Задача H. Посчитай в уме

Пусть $sum$ — сумма всех чисел на столе, $num$ — их количество, тогда зная эти два значения всегда можно подсчитать среднее арифметическое чисел на столе как $\frac{sum}{num}$. Давайте будем их хранить. Для этого создадим очередь в которой будут храниться все значения в том порядке, в котором их выложили, также пусть у нас будет словарь в котором ключ — это число на листочке, значение — количество таких чисел на столе. Также будем отдельно хранить сами значения $sum$ и $num$, оба изначально 0.

При появлении листочка с числом, будем увеличивать $sum$ на $x$, $num$ на 1, также будем добавлять $x$ в очередь и словарь.

При появлении листочка с «-» $num$ уменьшается на 1, $sum$ на $a$, где $a$ — первый элемент очереди, также уменьшаем значение словаря по этому ключу на 1 и удаляем его из очереди.

Для листочка с «?» будем считать $y = \frac{sum}{num}$, если $y$ – нецелое, то выводим 0, иначе смотрим сколько таких значений в словаре, это и есть ответ. Нетрудно заметить, что при таком моделировании все вышеперечисленное работает и работает быстро.

# Задача I. Сеть телепортов

Посмотрим на вершину запроса $v$ и подвесим дерево за неё. Теперь вершина $v$ — корень дерева, а значит все оставшиеся вершины — ее потомки. Посмотрим на какого-нибудь сына $u$ вершины запроса. Если в поддереве вершины $u$ есть вершина, которая находится на расстоянии большем $d$ от вершины $v$, то в этом поддереве точно надо удалить хотя бы одно ребро. Если все поддерево останется нетронутым, то эта вершина все еще будет достижима.

Поэтому, если в поддереве вершины $u$ есть «далекая» от $v$ вершина, удалим ребро $(v, u)$, тогда ни одна вершина из этого поддерева не будет достижима из вершины $v$, в частности, не будут достижимы вершины на расстоянии больше $d$. Если же в этом поддереве нет такой вершины, то и удалять рёбра из него не нужно, потому что поддеревья разных детей независимы между собой.

Ограничение по времени, правда, не позволяет для каждого запроса честно переподвешивать дерево за вершину из запроса, после чего проверять описанное выше свойство для всех ее детей. Подвесим дерево за вершину 1, после чего посчитаем для каждой вершины $\texttt{dp1}[v]$ — максимальную длина пути из $v$ в ее потомка, и $\texttt{dp2}[v]$ — максимальную длину пути из $v$, первое ребро которого ведет из $v$ наверх по дереву. Будем считать обе динамики с помощью *обхода в глубину*.

Первым обходом в глубину посчитаем `dp1` — для этого после того, как из вершины $v$ посещены все ее дети, отсортируем их по возрастанию `dp1`, после чего запишем

$$\texttt{dp1}[v] = \max_{u \in \text{children}(v)} \texttt{dp1}[u] + 1$$

.

Вторым обходом посчитаем dp2. Максимальный из путей «вверх» по ребру $v \to w$ либо идет «вверх» из $w$, либо идет из $w$ в ее поддерево, но **не по ребру** $w \to v$. Первое значение — это просто $\texttt{dp2}[w]$, а второе — максимальное из dp1 детей $w$, не равных $v$, плюс 1 (что можно найти за $\mathcal{O}(1)$, так как дети каждой вершины отсортированы по возрастанию dp1):

$$\texttt{dp2}[v] = \max\left(\texttt{dp2}[w], \max_{\substack{u \in \text{children}(w) \\ u \neq v}} \texttt{dp1}[u] + 1\right) + 1$$

Посчитав две такие динамики, можно за $\mathcal{O}(\log n)$ определять, по скольким ребрам из вершины достижимы вершины на расстоянии больше $d$: для пути «вверх» достаточно сравнить $\texttt{dp2}[v]$ и $d$, а для путей «вниз» — сделать бинпоиск по детям вершины, так как они отсортированы по dp1.

# Задача J. Придуманная проблема

Будем действовать жадно. Идя слева направо по первой строке, будем поддерживать текущую полученную сумму, и если она равна соответствующему значению из второй, то начинаем набирать следующее значение, иначе идем дальше. Если после перебора всех значений из первой строки получилось получить все цифры из второй и при этом не осталось лишних, то выводим «YES», иначе «NO».

# Задача K. Самодельный принтер

Таблица количества поднятий "руки"для каждой буквы:

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 2 | 2 | 1 | 3 | 3 | 2 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 1 |

# Задача L. Новая фигура

Несложно заметить, что фигура всегда ходит по одному цвету шахматной доски, так как передвигается на четное манхэттенское расстояние (манхэттенское расстояние - растояние от точки $(x_1, y_1)$ до точки $(x_2, y_2)$ вычисляемое по формуле $|x_1 - x_2| + |y_1 - y_2|$ ). Следовательно все, что нужно, это проверить, что манхэттенское расстояние от стартовой до другой клетки четно.