# Задача A. Цукерки

Формула - $a + b - c$.

# Задача B. Фрукти

Потрібно перевірити два твердження: $a + c \geqslant n \cdot x$ та $b + d \geqslant n \cdot y$.

# Задача C. Щасливі квитки

Для того, щоб отримати перше число, потрібно $n$ поділити на 100. Щоб отримати друге число, потрібно взяти залишок від ділення $n$ на 100. Після цього потрібно перевірити чи кожне з цих чисел дає залишок від ділення на певне число 0.

# Задача D. П'ятірка

Напишемо рекурсивну функцію $f(x)$, яка буде перевіряти, чи число містить 5.

Якщо $x = 0$, то повертаємо `false`. Якщо залишок від ділення $x$ на 10 - це 5, то значить ми знайшли п'ятірку та повертаємо `true`. Інакше ми повертаємо значення $f(\frac{x}{10})$.

# Задача E. Відбір на фінал

Спочатку видалимо усіх учасників, які народили раніше 2007-го року. Вони ніяк не будуть впливати на відповідь.

Відсортуємо учасників по першому турі. Додамо у список усіх учасників, у кого принаймні стільки ж балів, скільки у $k$-го. Видалимо їх усіх зі списку.

Повторимо алгоритм для другого та третього турів.

# Задача F. ML

Нехай $b_i = a_i - i + 1$. Тоді $a_i = b_i + i - 1$.

Тоді формула виглядатиме так:

$$\sum_{i=1}^{n} |a_i - (x + i - 1)|$$

$$\sum_{i=1}^{n} |(b_i + i - 1) - (x + i - 1)|$$

$$\sum_{i=1}^{n} |b_i + i - 1 - x - i + 1)|$$

$$\sum_{i=1}^{n} |b_i - x|$$

Можна помітити, що у цій формулі мінімум досягається, коли $x$ рівний медіаному елементу.

# Задача G. Кристали

Давайте ми спочатку будемо використовувати лише звичайні запуски, а потім лише посилені запуски.

Нехай ми використали $c_1$ звичайних запусків першого кристалу, $c_2$ другого, і так далі. Нехай $c_i > 0$, $c_j > 0$, $i \neq j$, $a_i < a_j$, тоді нам немає сенсу використовувати $i$-ий кристал у звичайному режимі, замість нього ми могли б використати кристал $j$ та отримати більше. Отже, є сенс використовувати лише один кристал у звичайному режимі - кристал з максимальним $a_i$. Нехай $a_m$ — максимальний $a_i$.

Але як зрозуміти, скільки разів нам потрібно його використовувати? У нас можуть бути кристали, у яких $b_i > a_m$. Тоді є сенс використовувати їх.

Відсортуємо $b_i$ за спаданням. Видалимо усі $b_i$, які менші за $a_m$. Будемо брати $b_i$ по черзі (у порядку спадання) доти, доки ми не отримаємо сумарно принаймні $k$. Якщо ж ми всі видалили, але цього недостатньо, то будемо тепер використовувати $a_m$ стільки разів, скільки потрібно. Нехай $x$ — це та потрібна кількість використань $a_m$, щоб отримати $k$.

Отже, ми тепер можемо спочатку використати кристал $m$ (той, у якого максимальне $a_i$) $x$ разів ($x$ може бути 0) у звичайному режимі, після цього використовувати кристали з максимальними $b_i$ у посиленому режимі.

# Задача H. Стовпчики

Можна помітити, що для $k = 0$, відповідь - це сума $a_1 + \sum_i \max(0, a_i - a_{i-1})$.

Якщо $k = 1$, то нам вигідно зробити так, щоб певне $a_i$ було рівне або $a_{i-1}$, або $a_{i+1}$. У такому випадку дві послідовні $a_i$ будуть однаковими, що є найбільш вигідним. Можна помітити, що при $k = 1$, задача фактично зводиться до того, що ми можемо видалити один елемент з $a$. Глянемо на приклад, нехай $a = [3, 1, 4]$. Якщо ми видалимо другий елемент, то буде $[3, 4]$. Якщо ми замінимо 1, наприклад, на 4, то вийде $[3, 4, 4]$. Тоді виходять два послідовні елементи, які фактично можна об'єднати в один.

Це саме можна сказати для будь-якого $k$. Задача зводиться до того, щоб видалити з масиву $k$ чисел.

Можемо використати метод динамічного програмування. Нехай $dp[i][j]$ - відповідь, якщо ми переглянули перші $i$ елементів зліва, взяли $i$-ий елемент, і взяли $j$ елементів сумарно. Формула виходить така:

$$dp[i][j] = \min_{k=1}^{i-1} dp[k][j-1] + \max(0, a_i - a_k)$$

Таке ДП можна порахувати за $O(n^3)$.

Задачу можна звести до того, щоб кожному $dp[i][j]$ присвоїти мінімум з

1. $dp[k][j-1]$, де $a_k > a_i$.
2. $(dp[k][j-1] - a_k) + a_i$, де $a_k \leqslant a_i$.

Для швидкого підрахунку будемо зберігати $2n$ дерев Фенвіка, половина з них буде відповідати за першу частину формули, а друга за другу. Звісно, тут також можна використовувати й дерева відрізків, проте потрібно сильно постаратися, щоб такий розв'язок не отримав ТЛ. Сумарна складність $O(n^2 \log n)$.