Задача A. Тура

Можна побачити, що відповідь — це $n + m - 2$.

Задача B. Координати

В цій задачі достатньо підставити значення в формулу. Варто замітити, що $\left(\sqrt{x}\right)^2 = x$, отже $\left(\sqrt{(x-0)^2 + (y-0)^2 + (z-0)^2}\right)^2 = x^2 + y^2 + z^2$.

Задача C. Улюблена іграшка Сакурако

Автор задачі: **Ціцей Павло**
Задачу підготував: **Ціцей Павло**
Розбір написав: **Ціцей Павло**

Помітимо, що після кожного кроку Сакурако відстань до іграшки зменшується максимум на k. Тобто, нам потрібно визначити, скільки разів відняти від m число k, щоб воно стало недодатнім (тобто від'ємним або нуль). Тобто, відповідь буде $\lceil \frac{m}{k} \rceil$.

Задача D. Не улюблена іграшка Сакурако

Автор задачі: **Ціцей Павло**
Задачу підготував: **Ціцей Павло**
Розбір написав: **Ціцей Павло**

Помітимо, що після кожного ходу, відстань від Сакурако до іграшки змінюється з m до $m-n+q$ (якщо Сакурако зробить крок розміру n). Будемо вважати, що Сакурако кожен раз робить крок розміру k, тоді відстань від Сакурако до іграшки буде зменшуватися по $k - q$ кожен раз, з чого зрозуміло, що якщо $k \leqslant q$, то відповідь -1, а інакше ця задача зводиться до задачі C, а тому відповідь буде $\lceil \frac{m}{k-q} \rceil$.

Задача E. Успішний інвестор

Автор задачі: **Тимкович Олександр**
Задачу підготував: **Тимкович Олександр**
Розбір написав: **Тимкович Олександр**

Будемо використовувати жадібний алгоритм. Ми знаємо, що b_2 повинно бути більшим за b_1, але немає ніяких нерівностей для b_1, то чому б нам b_1 не зробити мінімальним можливим, тобто $a_1 - x$.

Припустимо, ми вже жадібно знайшли всі значення $b_1, b_2, \ldots, b_{i-1}$ і тепер потрібно знайти b_i. Будемо його заміняти на мінімальне можливе значення так, щоб всі умови досі виконувались.

У кінці, достатньо перевірити, чи $b_1 < b_2 < \cdots < b_n$.

Задача F. Різдвяний вечір

Автор задачі: **Тимкович Олександр**
Задачу підготував: **Ціцей Павло, Тимкович Олександр**
Розбір написав: **Тимкович Олександр**

Для зручності, зробимо індексацію з нуля, тобто $a_0, a_1, \ldots, a_{n-1}$. Таке перетворення досить зручно в задачах з циклічними масивами, тому що ми можемо брати сусідні значення як $(a_i, a_{(i+1) \bmod n})$.

Нехай розчарування всіх друзів, не включаючи Сакурако, це S, що можна порахувати на перед. Будемо перебирати позицію Сакурако, тобто кожну пару сусідніх друзів і Сакурако між ними. Як зміниться S, якщо ми туди всунемо Сакурако? Потрібно від S відняти розчарування, яке було між цією парою друзів, та додати два нові розчарування, отримані з Сакурако.

Задача G. Чефір загубився

Автор задачі: **Ціцей Павло**
Задачу підготував: **Фейса Богдан**
Розбір написав: **Фейса Богдан**

У цій задачі будемо розглядати кучу випадків. Треба помітити, що за допомогою операцій, n тільки зменшується.

- $x \equiv n \equiv 0 \bmod 2$. У такому випадку можна виконувати операцію 1, тобто віднімати від n найменший дільник, який більший за 1. Бо n парне, то цей дільник буде 2, тому $n \to n-2$. При тому, парність n не змінилась. Оскільки x також парне, то рано чи пізно такими операціями ми зробимо з n число x.
- $x \equiv 0 \bmod 2$ та $n \equiv 1 \bmod 2$. Будемо зводити цей випадок до першого. Зробимо знову операцію 1. Зверніть увагу, що дільник, який ми віднімемо від n, буде непарним, бо якби він був парним, то n б ділилось на 2, але n непарне.
- $x \equiv 1 \bmod 2$ та $n \equiv 0 \bmod 2$. Спочатку, зведемо n до $2x$ за допомогою першого випадку. Після цього, найбільший дільник n буде $\frac{n}{2}$, бо $n = 2x$, тому використовуємо другу операцію і зводимо до x. Також, тут треба розглянути випадок, коли $n < 2x$ та вивести "`NO`".
- $x \equiv n \equiv 1 \bmod 2$. Віднімаємо від n найменший дільник, який буде непарним, та повторюємо алгоритм з третього способу.

Задача H. Сакурако та падаюче дерево

Автори задачі: **Фейса Богдан, Тимкович Олександр**
Задачу підготував: **Фейса Богдан**
Розбір написав: **Фейса Богдан**

На повний бал задача розв'язується за допомогою динамічного програмування по дереву.

Нехай $dp[v][0]$ відповідає за мінімальну кількість вершин з піддерева v, які нам потрібно взяти для того, щоб **кожна** іграшка в піддереві v була приєднана до якоїсь іншої іграшки з цього ж піддерева.

Також нехай $dp[v][1]$ відповідає за мінімальну кількість вершин з піддерева v, які нам потрібно взяти для того, щоб якась іграшка в піддереві v не була приєднана до якоїсь іншої іграшки з цього ж піддерева і її потрібно було спарувати з якоюсь іграшкою поза цим піддеревом.

Тоді, для листків нашого дерева значення будуть наступні:

$$dp[v][0] = \begin{cases} INF, \text{якщо в листку знаходиться іграшка} \\ 0, \text{якщо листок порожній} \end{cases}$$

$$dp[v][1] = \begin{cases} 1, \text{якщо в листку знаходиться іграшка} \\ INF, \text{якщо листок порожній} \end{cases}$$

Для вершини визначимо наступні значення:

$mn-$ сума мінімальних значень для піддерев вершини, якщо для кожної дитини дозволено вибрати лише $dp[i][0]$

$mn1-$ сума мінімальних значень для піддерев вершини, якщо для кожної дитини дозволено вибрати $dp[i][0]$ або $dp[i][1]$ але як мінімум один раз вибирається $dp[i][1]$

$mn2-$ сума мінімальних значень для піддерев вершини, якщо для кожної дитини дозволено вибрати $dp[i][0]$ або $dp[i][1]$ але як мінімум 2 рази вибирається $dp[i][1]$

Для того, щоб визначати переходи, розіб'ємо все на 2 випадки:

Якщо в вершині міститься іграшка, то переходи наступні:

$$dp[v][0] = mn1 + 1$$

$$dp[v][1] = min(mn1, mn) + 1$$

Якщо ж іграшки нема, то переходи наступні:

$$dp[v][1] = mn1 + 1$$

$$dp[v][0] = min(mn2 + 1, mn)$$

Всі ці значення можна обрахувати за сумарний час $O(n)$.