A. Повний чіназес

У цій задачі потрібно підтримувати скільки грошей було у Зеника після кожної операції та перевірити чи було це значення від'ємним хоча б 1 раз.

B. Менші зліва, менші справа

Якщо ми знаємо, скільки менших значень повинно бути зліва і скільки менших справа, то їх сума – кількість менших значень всього в масиві.

Відповідно, якщо $l_i + r_i = l_j + r_j$, то відповідні значення теж мають бути однакові. А якщо $l_i + r_i > l_j + r_j$, то значення на позиції i має бути більшим за значення на позиції j.

Масив $a_i = l_i + r_i$ задовольняє ці умови. Оскільки відповідь завжди існує, то можна вивести цей масив.

C. Потрійний XOR

Розглянемо i-тий біт. Якщо всі числа, які ми візьмемо в набір матимуть цей біт включений, то неможливо буде вибрати три числа з ксором 0, оскільки в ксорі цей біт входитиме тричі. Проте, після того, як ми взяли всі числа з i-тим бітом ми можемо взяти до множини ще щось.

Нехай ми хочемо взяти числа в яких старший біт буде або i-тий або j-тий. Нехай $i > j$. Тоді ми можемо набрати в множину всі числа з j-тим бітом і всі числа з i-тим бітом, в яких j-тий біт виключений. Можна показати, що це і буде оптимальний спосіб обрати числа. Можна показати, що в оптимальній відповіді $j = i - 1$.

Відповідно переберемо i. Візьмемо всі числа з проміжку $[2^{i-1}, 2^i + 2^{i-1})$, які належать нашій множині.

Складність розв'язку $O(n)$.

D. Лабіринт

Використаємо метод динамічного програмування. Для кожного орієнтованого ребра (v, p), де p — батько вершини v, порахуємо математичне очікування часу, необхідного для переходу з вершини v до вершини p. Тоді відповідь на задачу для будь-якої вершини v - це сума на вертикальному шляху від v до 1.

$dp[v, p]$ можна порахувати наступним чином. Нехай $t_1, t_2, ..., t_k$ - сини вершини v, посортовані за зростанням значення $dp[t_i, v]$. Тоді $dp[(v, p)] = 1 + 2^{-k} + \sum_{i=1}^{k}(1 + dp[t_i, v]) \cdot 2^{-i}$.

E. Трикутники на площині

Відсортуємо всі $3 \cdot n$ точок за зростанням їхніх координат x. У випадку рівності координат x, відсортуємо за зростанням координат y. Таким чином, отримаємо впорядкований список точок: $P_1, P_2, \ldots, P_{3n}$.

Розділимо відсортований список точок на n груп по три сусідні точки:

$$\{P_1, P_2, P_3\}, \{P_4, P_5, P_6\}, \ldots, \{P_{3n-2}, P_{3n-1}, P_{3n}\}$$

Для кожної групи з трьох точок побудуємо трикутник. Таким чином, отримаємо n трикутників, що задовольняють умову.

F. Обміни з максимальною сумою

Нехай ми хочемо замінити i елементів з лівої частини на i елементів з правої. Відповідь на задачу тоді буде: сума в лівій частині - сума i елементів, які ми заміняємо в лівій частині + сума i елементів, які ми заміняємо в правій частині. Можна показати, що однією з оптимальних послідовностей операцій буде:

- Посунути i елементів зліва, які ми збираємося міняти в найправіші i позицій лівої частини. Тобто, поставити їх на позиції $n - i$, $n - i + 1$, $\ldots$, $n - 1$.

- Посунути i елементів справа, які ми збираємося міняти в найлівіші i позицій правої частини. Тобто, поставити їх на позиції n, $n+1$, $\ldots$, $n+i-1$.
- Виконати $i \cdot i$ операцій, щоб свапнути елементи.

Порахуємо динаміку $dp1_{i,j}$ — мінімальна сума i елементів в лівій частині, якщо ми зробили j ходів, щоб виконати перший крок. Аналогічно порахуємо динаміку $dp2_{i,j}$ — максимальна сума i елементів в правій частині, якщо ми зробили j ходів, щоб виконати другий крок. Третій крок зробити просто. Таку динаміку можна рахувати за (n^4).

Об'єднати такі дві динаміки можна за $O(n^5)$, але це повільно. Модифікуємо трохи другу динаміку наступним чином: $dp_{i,j}$ — Нехай ми зробили перший крок і виконуємо другий. Нам треба ще i елементів досунути в правій частині та ми зробили вже сумарно j кроків. Ми будемо її перераховувати за допомогою $dp1$. Таку динаміку можна рахувати за $O(n^4)$ і вона зразу покриє перші два кроки. Оскільки нам треба зробити щонайбільше k кроків, то треба взяти ще максимум на префіксі.

G. Прямокутник і точки

Розглянемо мінімальну відповідь, якщо лівий нижній кут зсувається вгору та вправо. 3 інші випадки розв'язуються так само, за допомогою віддзеркалення точок.

Посортуємо всі точки по x, а при рівності по y. Йтимемо зліва направо, розглянемо точку p_i, яка має максимальну y координату серед всіх точок на суфіксі. Знайдемо такий найправіший індекс j, що $p_j.x \le p_i.x$ і $p_j.y \ge p_i.y$. Тоді якщо перемістити прямокутник так, що його лівий нижній кут був у точці $(p_j.x, p_i.y)$, то всі точки опиняться на межах або поза межами прямокутника. Назвемо таку точку $(p_j.x, p_i.y)$ цікавою.

Цікаві точки можна легко знаходити за допомогою стека, де завжди виконується, що остання точка є найправішою та нийнижчою серед усіх у стеці, а якщо ми намагаємося додати точку, яка суперечить цьому, видалятимемо зі стека останню точку, поки умова не виконуватиметься. Порахуємо відповідь для всіх цікавих точок та виберемо мінімальну.

H. Доставка їжі

У цій задачі потрібно було вивести $a + c$, якщо $a < b$. Якщо ж $a \ge b$, то вивести a.

I. Перший етап

У цій задачі потрібно було перевірити усі задані умови. Найпростіше це було зробити, якщо спочатку пройтись по топ m команд, підтримуючи, скільки команд з кожного університету уже розглянуто.

Після цього можна пройтись по всіх командах поза топ m та додати по 1 команді кожного регіону, який ще не було розглянуто.

J. Гільдія злодіїв

Зробимо двійковий пошук по відповіді. Нехай наразі потрібно перевірити, чи відповідь є не більшою за x.

Будемо рахувати для вершини v, чи можливо орієнтувати піддерево вершини v. Для кожного сина to важливо, чи ми орієнтуємо ребро від v до to чи навпаки.

Якщо ми орієнтуємо ребро від v до to, то нам важливо піддерево to орієнтувати так, щоб для нього виконувалась умова, а також мінімізувати кількість монет досяжних з вершини to. Назвемо таке значення $down_{to}$.

Якщо ми орієнтуємо ребро від to до v, то також потрібно, виконувалась умова, а також мінімізувати найбільшу кількість монет досяжних з певної вершини, з якої також досяжна вершина to. Назвемо таке значення up_{to}.

Нехай A – це множина синів вершини v таких, що ми орієнтуємо ребро від неї до v, B - множина синів, де ми орієнтуємо ребро до сина.

Тоді нам потрібно розбити на множини так, щоб $max_{to \in A} up_{to} + \sum_{to \in B} down_{to} + a_v \leq x$.

Якщо посортувати всіх синів по значенню up, то вигідно деякий префікс взяти в множину A і суфікс в множину B. Так ми зможемо перевірити, чи все ще виконується умова. Залишилось перерахувати значення up_v та $down_v$.

up_v - мінімальне можливе значення $max_{to \in A} up_{to} + \sum_{to \in B} down_{to} + a_v$.

$down_v$ - мінімальне можливе значення $\sum_{to \in B} down_{to} + a_v$, серед таких, що задовольняють умову.

Тож обидва ці значення можна порахувати разом з перевіркою умови двійкового пошуку.

Складність розв'язку $O(n \log^2 n)$.

K. Зростаюча таблиця

Відсортуємо всі числа в порядку зростання. Спробуємо закладати матрицю діагоналями, тобто розглядатимемо клітинки в порядку зростання $i + j$. Клітинки з однаковим значенням $i + j$ потрібно розглядати або в порядку зростання i, або в порядку спадання. Переберемо два варіанти. Розставимо всі числа від менших до більших в клітинки в такому порядку. Перевіримо чи все виконується. Якщо хоча б для одного варіанту все виконалося, то виведемо відповідь. Інакше, відповідь `NO`.

L. Лови

Якщо перший гравець може піти в якомусь з напрямків і досягнути безпечної точки швидше за другого гравця, то він точно може перемогти.

Така умова не виконується лише, якщо обидва гравці початково знаходяться в одній чверті, на одній діагоналі, та другий гравець ближче до осей, ніж перший. Тобто, якщо $x_1 > 0$ та $y_1 > 0$, то повинно виконуватись $x_2, y_2 > 0$, $x_1 - y_1 = x_2 - y_2$, $x_1 < x_2$ та $y_1 < y_2$.

В такому випадку можна показати, що переможе другий гравець. Якщо перший гравець ходить в бік від осей, то другий гравець повторює його хід. Якщо ж до осей, то другий гравець робить хід, щоб залишитись на тій же діагоналі, але ближче до першого гравця.