# Гомельська обласна олімпіада 2018 рік (1 тур)

## Задача A, Два квадрата

Для зручності пронумеруємо клітинки полотна. Перший рядок пронумеруємо зліва направо числами від 1 до $n$, другий - від $n+1$ до $2n$, і так далі. Знайдемо зафарбовану клітинку з найменшим номером. Ця клітинка повинна бути лівим верхнім кутом одного з квадратів (ніякою іншою частиною квадрату вона бути не може, адже згори та зліва від неї всі клітинки білі). Аналогічно, клітинка з найбільшим номером повинна бути правим нижнім кутом іншого квадрату. Це твердження справедливе і для випадку, коли квадрати співпадають.

Залишилось знайти довжину сторони квадратів.

Повернемося до зафарбованої клітинки з найменшим номером (лівого верхнього кутка першого квадрату). Заведемо два вказівники та будемо одночасно рухати їх з цієї клітинки вправо та вниз, доки хоча б один вказівник не вийде за межі холсту або не потрапить до білої клітинки. Відстань, яку пройдуть вказівники, і буде довжиною сторони квадрату. Зауважимо, що рухати лише один вказівник недостатньо, адже два квадрати можуть бути намальовані поруч, що продовжить шлях одного з вказівників.

Знаючи довжину сторони квадратів, лівий верхній куток першого та правий нижній другого, можемо вивести відповідь.

# Задача B, Творческие выходные

Давайте для кожного кольору порахуємо скільки планок у паркані в нього пофарбовано (помітимо, що для якихось кольорів це значення може дорівнювати 0), який назвемо $count[a_i]$.

Нам необхідно придумати алгоритм фарбування паркану, керуючись яким, Тринідад Ітобагович витратив би якнайбільше часу. Так як він завжди працює лише з невикористаними банкам, то слід зауважити, що пофарбувавши і-ту планку ваі-ий колір, ми не зможемо її більше пофарбувати в інший колір (інакше не зможемо отримати кінцеву розфарбовку паркану). З цього слідує, що кожною новою банкою фарби ми можемо фарбувати всі планки і, що ще не пофарбовані в колір $a_i$.

Оптимальніше за все буде скористатися таким жадібним алгоритмом — фарбувати паркан (його доступні планки) у різні кольори в порядку зростання їх $count[a_i]$. Чому? Очевидно, що з кожною новою банкою фарби кількість планок у паркані, що що не пофарбовані у свій кінцевий колір, буде зменшуватися (як наслідок і кількість часу витраченого на фарбування планок), тому ми будемо використовувати фарби в такому порядку, щоб дана кількість зменшувалась найповільніше (у порядку зростання $count[a_i]$).

Складність такого розв'язку – $O(n \log n)$.

# Задача C, Непростая сумма

Скористаємось сортуванням підрахунком. Відповідь будемо рахувати як

$$\sum_{i=1}^{n} sum_i$$

, де

$$sum_i = cnt_i \cdot \sum_{j=1}^{n} (j\%i) \cdot cnt_j$$

, а $cnt_i$ - кількість чисел, що дорівнюють $i$. Залишилося навчитися ефективно підраховувати $sum_i$.

Помітимо, що всі числа, що менші за $i$, дають у якості залишку своє власне значення, тому нам достатньо підрахувати суму всіх чисел, що менші за $i$. З більшими числами вчинимо хитріше. Розіб'ємо їх на такі відрізки: $[i, 2i), [2i, 3i), [3i, 4i)$, ... . Навчимося швидко знаходити відповідь для таких відрізків.

Розглянемо відрізок $[k \cdot i, (k+1) \cdot i)$. Помітимо, що число $k \cdot i + j$ при діленні на $i$ дає залишок $j$. Тому, якщо ми присто додамо до відповіді число $k \cdot i + j$, то відповідь збільшиться на $k \cdot i$ більше, чим треба. Тому, якщо ми складемо всі числа з відрізку $[k \cdot i, (k+1) \cdot i)$, то нам потрібно буде прибрати $(k \cdot i) \cdot cnt_{k \cdot i \ldots (k+1) \cdot i}$. Через це суму залишків на такому проміжку можна обчислити як $sum_{k \cdot i \ldots (k+1) \cdot i} - (k \cdot i) \cdot cnt_{k \cdot i \ldots (k+1) \cdot i}$.

Такий розв'язок потребує $O(n)$ часу на сортування і $O(n + \frac{n}{2} + \frac{n}{3} + \ldots + \frac{n}{n}) = O(n \log n)$ на обчислення відповіді.

# Задача D, Роборалли

Розглянемо орієнтований граф у котрому є нескінченна кількість шляхів (у ньому завжди знайдуться цикли) та на кожному ребрі є певний символ, котрий ми отримаємо перейшовши по цьому ребру. Якщо починаючи з вершини $X$ ми вже маємо стрічку довжини $len$, то можемо почати шлях (однозначний - оскільки, за умовою, з кожної вершин виходить лише одне ребро) ми можемо на кожному кроці отримувати новий символ та утворювати нову стрічку довжини $len+1$. Таким чином ми для графу з $N$ вершин ми можемо отримати саме $N$ стрічок заданої довжини і для кожної вершини вони будуть однозначні.

Якщо ми хочемо отримати стрічку довжини $M$, то починати з кожної вершини та утворювати стрічку такої довжини для порівняння досить неоптимально. Такий розв'язок буде працювати $O(N \cdot M)$.

Розглянемо метод двійкового підйому.

Якщо склеїти два ребра довжини 1, то вийде одне ребро довжини 2. А якщо склеїти 2 ребра довжини 2 ($a \to b$ і $c \to d$), то можна також отримати 1 ребро довжини 4 $a \to d$. І так далі. Тому на кожному кроці замість порівняння рядків, що утворюють шляхи довжини $2^x$, можна порівняти лише вершини та утворити шлях довжини $2^{x+1}$.

Існує декілька варіантів підтримки відсортованості шляхів. Пропонуємо наступний. Для кожного такого об'єднання відсортуємо всі шляхи довжини $2^x$ та порівняємо символи, що їх презентують; на кожному кроці кожному шляху надамо певний порядковий номер, який не може покращитись зі збільшенням значення $X$, а може лише погіршитись (якщо на черговому кроці ми поєднаємо два шляхи, які презентують різні символи, хоча до певного етапу вони були однакові, а якщо на етапі $X$ презентуючий символ "погіршився" для вершини $A$, то вона втратить свою позицію у масиві відсортованих шляхів).

У цій задачі для простоти можна вирішити її більше як для числа $M$, а для числа $2^x \geq M$.