Algorithmic Leo Camp

День 3

Розбір задач

F - Best Website

- Оскільки обмеження дуже малі, то можна перебрати перший символ входження “algotester” в Зениковий нікнейм та перевірити чи стрічка може починатися у цій позиції.
- Якщо хоч один раз шукана стрічка зустрілась, виводимо “Yes”, інакше - “No”.

H - Morning Run

- Швидший бігун доганяє повільнішого.
- Якщо швидша Марічка, то їй треба скоротити відрив у D метрів, а якщо Зеник, то йому - у $(L - D)$ метрів.
- Швидкість, з якою швидший бігун скорочує відрив - $|Z - M|$.
- Час, за який швидший бігу дожене повільнішого, рівний відстані відриву поділеній на швидкість скорочення відриву.
- Складність: $O(1)$.

B - Decrease array

- В кінцевому масиві всі елементи рівні. Інакше ми можемо зробити ще один крок.
- Операція не зміню НСД масиву.
- Отже, в результаті всі елементи масиву будуть рівні НСД.
- Алгоритм Евкліда знаходить НСД двох чисел з логарифмічною складністю.
- *НСД(a, b, c) = НСД(НСД(a, b), c).*
- Сумарна складність: $O(N \log(maxA_i))$.

D - Vaccination

- Уявимо, що ймовірності вакцин фіксовані - $P_1, P_2, \ldots, P_N$.
  - Тоді відповідь - $(1 - (1-P_1)(1-P_2)\ldots(1-P_N))$.
- Замінимо рівномірно розподілені ймовірності математичним сподіванням: $P_i = (L_i + R_i) / 2$.
  - Зауважте, що L_i та R_i потрібно попередньо поділити на 100, щоб перейти від відсотків до ймовірностей.
- Складність: $O(N)$.

L - Construct Array

- Використовуватимемо лишень числа *1, 2 та 3.*
- Одиниця утворює просту суму лише з іншою одиницею чи двійкою.
- Масив з *K* одиниць містить *F(K) = K(K-1)/2* простих сум.
- Масив з *K* одиниць та двійки містить *F(K+1)* простих сум.
- Трійки не утворюють простих сум між собою та з одиницями, проте утворюють з двійкою.
- 1....1 2 3...3 (*K* одиниць, *M* трійок) містить *F(K+1)+M* простих сум.
- Знайдемо таке найбільше *X*, що *F(X+1) ≤ N*.
    - *X та N - F(X+1)* є порядку кореня з *N.*
- 1...1 2 3...3 (*X* одиниць*, N-F(X+1)* трійок) має рівно *N* простих сум.
- Складність: *O(sqrt(N))*.

E - Lazy Game

- Друга частина - відома гра - Нім.
  - Другий гравець програє, якщо xor-сума розмірів купок = 0.

- Для будь-якого парного *x*: $x \oplus (x+1) \oplus (x+2) \oplus (x+3) = 0$.
  - Якщо *L* парне та $L+3 \le R$, то ми можемо вибрати 4 купки з сумарним розміром $4*L+6$ (*L*, *L + 1*, *L + 2* та L + 3).
  - Якщо *L* непарне та $L+4 \le R$, то ми можемо вибрати 4 купки з сумарним розміром $4*L+10$ (L + 1, L + 2, L + 3, L + 4).
- Чи можна обійтися менше ніж чотирма купками ?
- Менше ніж трьома точно не можна.

E - Lazy Game

- Нехай *len(x)* - довжина двійкового запису *x* без ведучих нулів.
- Припустимо, що існує відповідь з трьох купок $a < b < c$:
  - $len(a) < len(b) = len(c)$, тоді $len(c) >= len(a) + 1$
  - Рівно в одному з чисел *b* та *c* має бути встановлений старший біт числа *a*.
  - Мінімальне значення для *c*: $2(2^{len(a)-1}) + (2^{len(a)-1})$.
  - Отже, $c = 3*2^{len(a)-1}$, $b = a \oplus c$.
  - Якщо *a* зростає, то *b* та *c* неспадають.
  - Отже, відповідь з трьох купок існує якщо $X = 3*2^{len(L)-1} <= R$ і мінімальна сума розмірів купок в цьому випадку рівна $L+X+(L \oplus X)$.
- Складність: $O(log(L))$ на запит.

G - Divisible Array

- Переберемо останній елемент масиву - нехай це *m.*
- Розглянемо представлення *m* у вигляді добутку простих.
- Розв'язуватимемо по кожному простому незалежно
  - Нехай просте число *p* входить у *m* рівно *t* разів
  - Зрозуміло, що для всіх $i \in [1, n]$ $w(p, a_i) \geq w(p, a_{i-1})$, де $w(p, x)$ - це кількість входжень числа *p* в *x* (вважатимемо $a_0 = 0$)
  - Тому кількість різних масивів *A* для числа *p* буде рівна кількості послідовностей з *k* невід'ємних чисел з сумою *t* - C_{k+t-1}^{k-1}
  - Таку величину можна рахувати за *O(t).*
- Решетом Ератосфена знайдемо найменший простий дільник для кожного числа, що не перевищує *n.*
- Сумарна складність - *O(n log log n).*

J - Piles of stones

- В кожному елементі починається не більше одного відрізка.
- В кожному елементі закінчується не більше одного відрізка.

- Коли відповідь рівна нулю?
  - Існує елемент більший за *K*.
  - Різниця між двома сусідніми елементами > 1.
  - Перший чи останній елемент < *K-1.*

J - Piles of stones

- Нехай C - кількість таких i, що $A_i = A_{i+1}$ та $A_i < K$. Для кожного i є два варіанти:
    - A_{i+1} покривається точно тими ж відрізками, що й A_i.
    - В A_{i+i} закінчується найлівіший з відрізків, що покривають A_i, та починається новий.
- Все решта визначається однозначно.
- Якщо відповідь не нуль, то вона - 2^C.
- Складність: $O(N)$.

A - King Zenyk

- Деякі клітинки поля 100x100 потрібно заблокувати, аби розміри компонент зв'язності при обході поля конем були рівні заданому набору.
- Будуватимемо поле так, щоб всі ці компоненти були шляхами.
- Мета - знайти два шляхи довжини принаймні 1000, щоб кожна клітинка у них мала не більше двох незаблокованих сусідів.
- Коли ми їх знайдемо, то будемо ставити перегородки на шляхах, аби отримати потрібні розміри компонент.
- Зважаючи на умову того, що сума розмірів всіх компонент не перевищує *1000* це завжди можливо.

A - King Zenyk

	1	1	3	3		
		2	2	4	4	
					5	5
		8	8	6	6	
	9	9	7	7		
10	10					

I - Number Coloring

- Розглянемо лиш ті числа, які є степенями двійки
- Спробуємо зафарбувати їх двома кольорами.
  - $2 \to 4 \to 16 \to 8 \to 32$
  - Для $n > 31$ розфарбування у два кольори неможливе.
- Можна показати, що числа 2, 4, 8 ..., 2^{13} можливо пофарбувати у три кольори, проте 2, 4, 8 ..., 2^{14} вже ні.
  - Шукане розфарбування можна знайти перебравши всі 3^{13} розфарбувань та перевіривши кожне з них.
- Тепер для довільного *x* пофарбуємо його у той же колір, що і число $2^{\omega(x)}$, де ω(x) - кількість простих дільників *x* *(враховуючи кратності)*.
- Таке розфарбування буде коректним, адже у кожній групі числа матимуть однакову кількість простих дільників, а тому добуток двох з них матиме більшу кількість простих дільників ніж будь-яке третє.

C - Fibonacci Matrix

- Зробимо прості перетворення

$$f_{i+1}^2 \cdot M^{i+1} = (f_i + f_{i-1})^2 \cdot M^{i+1} = (f_i^2 + 2 f_i f_{i-1} + f_{i-1}^2) \cdot M^{i+1}$$

$$f_{i+1} f_i = (f_i + f_{i-1}) f_i = f_i^2 + f_{i-1} f_i$$

- Представимо у матричному вигляді
    - Кожен елемент - матриця. 1 - одинична матриця, 0 - нульова матриця.

$$\begin{pmatrix} f_{i+1}^2 M^{i+1} \\ f_i^2 M^i \\ f_{i+1} f_i M^{i+1} \end{pmatrix} = \begin{pmatrix} M & M^2 & 2M \\ 1 & 0 & 0 \\ M & 0 & M \end{pmatrix} \times \begin{pmatrix} f_i^2 M^i \\ f_{i-1}^2 M^{i-1} \\ f_i f_{i-1} M^i \end{pmatrix}$$

C - Fibonacci Matrix

- Додамо суму

$$\begin{pmatrix} f_{i+1}^2 M_{i+1} \\ f_i^2 M_i \\ f_{i+1} f_i M^{i+1} \\ \sum_{j=0}^{j} f_j^2 M^j \end{pmatrix} = \begin{pmatrix} M & M^2 & 2M & 0 \\ 1 & 0 & 0 & 0 \\ M & 0 & M & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} f_i^2 M_i \\ f_{i-1}^2 M_{i-1} \\ f_i f_{i-1} M^i \\ \sum_{j=0}^{i-1} f_j^2 M^j \end{pmatrix}$$

$$\begin{pmatrix} f_{k+1}^2 M_{k+1} \\ f_k^2 M_k \\ f_{k+1} f_k M^{k+1} \\ \sum_{j=0}^{k} f_j^2 M^j \end{pmatrix} = \begin{pmatrix} M & M^2 & 2M & 0 \\ 1 & 0 & 0 & 0 \\ M & 0 & M & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}^k \times \begin{pmatrix} M \\ 1 \\ M \\ 1 \end{pmatrix}$$

C - Fibonacci Matrix

- Скористаємося бінарним піднесенням до степеня.
- Складність: $O(4^3\ n^3 log(k))$.

K - Coronavirus polynomial

- Базис Ньютона для цілочисельних многочленів

$$P(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} c_i C_x^i$$

- Знайдемо коефіцієнти в цьому базисі

$$c_0 = f(0), c_1 = f(1) - c_0, c_2 = f(2) - c_0 - 2c_1$$

$$c_k = f(k) - \sum_{i=0}^{k-1} C_k^i c_i$$

K - Coronavirus polynomial

- Розглянемо приклад $P(x) = x^3 + 1$

$$P(0) = 1, P(1) = 2, P(2) = 9, P(3) = 28$$

	$P(0)$	$P(1)$	$P(2)$	$P(3)$
c_0	1	2	9	28
c_1	1	7	19	
c_2	6	12		
c_3	6			

- Знайти базис можна за *$O(n^2)$* операцій.
- Знаходити $c_i C_k^i$ можна аналогічно до алгоритму бінарного піднесення до степеня, роблячи додавання замість множення.