День 1

Розбір задач

B - Найменше число

- Число N складається із не більше як 10 цифр.
- Переберемо всі $\leq 2^{10}$ варіантів видалення цифр (рекурсивно або використовуючи бітові маски).
- Серед тих, які видаляють не більше ніж K цифр, залишають хоча б одну цифру, та не залишають ведучих нулів, оберемо найменше.
- Складність: $O(|N|)$.

A - Робот

- Кожну координату *x*, *y*, *z* можна розглядати окремо.
- Порахуємо кількість можливих кінцевих координат по *x*, *y*, *z*, позначимо *cx*, *cy*, *cz*.
- Загальна відповідь рівна *cx • cy • cz*.
- Як порахувати *cx*? Просимулюємо процес починаючи з *x = 0* та *x = n-1*. Абсолютна різниця між кінцевими координатами буде рівна *cx*.
- Складність: *O(кількість команд)*.

H - Щаслива основа

- Якщо n рівне 4 або 7, то відповідь - безліч.
- Основа системи числення повинна бути хоча б 5.
- Оскільки $\log_5(10^9)$ не перевищує 13, у числі буде не більше ніж 13 цифр.
- Переберемо усі варіанти щасливих чисел із не більше як 13 цифрами.
- Для кожного з них перевіримо, чи існує основна системи числення, в які це число рівне n.
- Чим більша основа, тим більше число.
- Отже, основу можна знайти бінарним пошуком.

D - Хобомобіль

- Нехай по кожному ребру можна рухатись у довільному напрямку.
- Побудуємо граф, де вершини - це міста, ребра - двосторонні дороги між ними.
- Найкоротший шлях від вершини *A* до *B* буде відповіддю.
- Найкоротший шлях найпростіше знайти алгоритмом пошуку в ширину (BFS).

D – Хобомобіль

- Що робити, коли напрям ребер чергується?
- Роздвоїмо кожну вершину: одна буде відповідати за вершину в непраний момент часу, інша – за парний.
- Із кожної з *2N* вершин проведемо ребра, куди можна рухатись, враховують напрямок у відповідний момент часу.
- Тепер задача звелась до аналогічної попередній, проте є два варіанти початку та кінця (у парний чи непарний момент часу).
- Знову використаємо BFS.
- Складність: *O(N+M)*.

I - Свято наближається

- Зауважимо, що якщо прийнято рішення побудовати K нових будинків між парою сусідніх існуючих, то їх варто розміщувати на рівній відстані один від одного.
- За такої умови найбільша серед відстаней мінімізується.
- Для кожної пару існуючих сусідніх будинків будемо пам’ятати, скільки нових було побудовано між ними (спочатку 0), і яка відстань між сусідніми всередині (спочатку d).
- Зауважимо, що ця відстань завжди буде раціональним дробом.

I - Свято наближається

- У цій задачі працює жадібний алгоритм.
- *M* раз виконаємо наступні кроки:
  - Знайдемо найбільшу відстань.
  - Побудуємо новий будинок, таким чином зменшивши відстань.
- Для оптимізації використаємо *set*, *priority_queue*, або аналогічну структуру даних.

E - Щасливі числа Діда Мороза

- Навичмося знаходити к-ть щасливих чисел на проміжку *0..N*, та позначимо як *F(N)*. Тоді відповідь на задачу рівна *F(B) - F(A-1)*.
- Згенеруємо всі щасливі числа менші за 10^9.
- Серед них викинемо ті, які діляться на хоча б одне менше щасливе число (наприклад, 4444 ділиться на 44).
- ~~Зауважимо, що жодне число менше 10^9 не може одночасно ділитись на два щасливі числа (із нашого списку), які більші за 10^4, адже НСК будь-яких таких двох є більшим за 10^9.~~
- Також зауважимо, що кількість щасливих чисел, менших за 10^4, рівна 18.

E - Щасливі числа Діда Мороза

- Отже, можна рахувати окремо числа, які діляться на великі щасливі (від 10^4), проте не діляться на малі, та окремо ті, які діляться лише на малі (до 10^4).
- Потрібно все робити так, аби жодне число не порахувати двічі.
- Для кожного великого можна перебрати всі числа до 10^9, які на них діляться, адже ця кількість буде невеликою. **Зауважте, що є декілька чисел, які одночасно діляться на два великих щасливих числа. Отже, потрібно рахувати лише унікальні (наприклад, додаючи їх в set).**
- Для малих можна використати принцип включення-виключення.

G – Королівство Хобітів

- Будемо перебирати усі числа від більших до менших.
- Нехай поточне зафіксоване число буде мінімальним серед вибраного проміжку.
- Отже, у проміжок можуть попадати лише числа, які більше або рівні за зафіксоване (їх ми вже обробили).
- Будемо підтримувати структуру даних *DSU (disjoint set union)*.
- Усі числа, які ми вже розглянули та які є сусідніми, будемо об’єднувати.
- Таким чином, обробляючи поточне зафіксоване число ми знаємо розмір проміжку із не менших чисел, в якому він знаходиться.

G – Королівство Хобітів

- Нехай поточне зафіксоване число *X*, а розмір проміжку – *K*.
- Якщо *K* хоча б *A*, візьмемо рівно *A* чисел для мінімальної відповіді (тобто *X•A*), та рівно *min(B, K)* для максимальної (тобто *X•min(B, K)*).
- Серед усіх варіантів зафіксованого найменшого числа виберемо оптимальний окремо для мінімальної та максимальної відповіді.

C - Задача про дерево

- Використаємо алгоритм центроїдної докомпозиції.
- Виберемо центральну вершину та розіб’ємо дерево на шматки.
- Розв’яжемо кожне піддерево рекурсивно, і таким чином знайдемо пари шляхів, які повністю знаходяться в піддереві.
- Після цього потрібно порахувати всі шляхи, які знаходяться повністю або частково в різних шматках.
- Зафіксуємо шматок, в якому повністю знаходиться шлях.
- Порахуємо всі варіанти розміщення другого шляху.

F – Послідовність нулів

- Можна показати, що задача зводиться до наступної: впорядкувати послідовність чисел так, аби $K + A_1 + A_n + sum(max(A_i, A_{i+1})) \le N$.
- Будемо будувати послідовність, починаючи з пустою, додаючи числа в порядку зростання.
- Будемо вважати, що однакові числа є різними. Вкінці лише поділимо відповідь на факторіали по кожній групі однакових.

F - Послідовність нулів

- Використаємо динамічне програмування зі станом *[i][j][sum]*, де *i* - поточне число, яка обробляємо, *j* - кількість зафіксованих пар сусідніх, *sum* - сума максимумів сусідніх по всіх зафіксованих парах.
- Зафіксовану пару сусідніх елементів будемо називати таку, між якими більше не будемо вставляти елементи. Отже, їх максимум можна відразу додавати до *sum*.

F - Послідовність нулів

- Обробляючи поточний *i*-й елемент, тобто вставляючи його в послідовність, є декілька варіантів:
  - вставити на початок послідовності
  - вставити на початок та зафіксувати пару
  - вставити в кінець
  - вставити в кінець та зафіксувати пару
  - вставити всередину між будь-якими незафіксованими
  - вставити всередну між будь-якими незафіксованими за зафіксувати пару зліва
  - вставити всередну між будь-якими незафіксованими за зафіксувати пару справа
  - вставити всередну між будь-якими незафіксованими за зафіксувати пари зліва та справа

F - Послідовність нулів

- Для кожного із цих варіантів можна перерахувати новий стан, у який потрібно перейти.
- Зауважте, коли після вставлення i-го елементу він фіксується в парі з сусіднім, цей сусідній завжди є меншим або рівним, адже ми обробляємо числа в порядку неспадання.
- Відповідь потрібно шукати у станах, в яких $j = 0$ (тобто усі пари були зафіксовані і додані до *sum*).
- Складність: $O(NK^2)$.