

# Зміст

Ідея розв'язку задачі А	2
Ідея розв'язку задачі В	3
Ідея розв'язку задачі С	4
Ідея розв'язку задачі D	5

## Ідея розв'язку задачі А

Запропонував: Рубаненко Р.

Підготував: Міхно М.

Давайте спробуємо знайти вартість найменшого інгредієнту. Очевидно, це буде найменше число вхідного масиву, адже найменше число не можна отримати як суму деяких інших чисел. Як знайти інші елементи? Давайте припустимо, що ми встановили вартість найдешевших  $K$  елементів. Згенеруємо усі можливі суми підмножин цих  $K$  елементів та викреслимо ці суми із вхідного масиву. З елементів, що залишилися, візьмемо мінімальний. Він і буде наступним по вартості інгредієнтом.

Чому? Припустимо, на якомусь кроці ми оберемо не мінімальне число. Тоді, коли ми будемо генерувати усі суми підмножин для відповіді, ми ніколи не згенеруємо це мінімальне число, адже серед перших  $K$  чисел не буде такої підмножини (інакше мінімальне число було б викреслене на цьому кроці), а всі інші числа будуть більшими за поточний мінімум (ми візьмемо число, більше за поточний мінімум, і генеруємо послідовність-відповідь в порядку зростання).

До речі, з цього твердження випливає, що відповідь завжди встановлюється однозначно. Якщо для видалення елементів із вхідної послідовності використовувати звичайний масив, і видаляти елементи за  $O(\text{довжина масиву})$ , асимптотика буде складати  $O(n \cdot 2^n \cdot 2^n)$ , тому що ми робимо  $n$  ітерацій, на кожній видаляємо порядку  $2^n$  елементів за  $2^n$  операцій.

Якщо використовувати `set`, збалансоване бінарне дерево пошуку або дерево відрізків для видалення елементів, асимптотика поліпшиться до  $O(n \cdot 2^n \cdot n)$ , адже видалення тепер виконується за  $O(\log(2^n)) = n$ . Але числа в задачі невеликі (до  $10^6$ ), а тому можна використовувати масив підрахунку `cnt`, де `cnt[x]` зберігає кількість елементів  $x$ . Тоді видалення елементів буде виконуватись за  $O(1)$ , і загальна складність алгоритму стане  $O(n \cdot 2^n)$ , що дозволяє пройти останню групу тестів.

### **Асимптотика:**

По часу -  $O(n \cdot 2^n)$

По пам'яті -  $O(\alpha)$ , де  $\alpha$  - максимальна серед вартостей наборів.

## Ідея розв'язку задачі В

Запропонував і підготував: Баренблат І.

**Перш за все доведемо просту лему:**

Припустимо, що ми можемо обрати  $K$  мінйонів, задовольнивши при цьому умову задачі. Якщо викинути мінйона з найбільшою витривалістю, то середнє арифметичне витривалостей **не збільшиться**, а мінімальна з сил мінйонів **не зменшиться**. Таким чином, загін з  $K - 1$  мінйона ми також можемо утворити.

Далі у нас є кілька варіантів розв'язку:

### Варіант 1

Застосуємо принцип бінарного пошуку. За лемою, якщо можна утворити загін розміру  $K$ , то і загони меншого розміру утворити теж можна, тож залишилось лише навчитись перевіряти можливість створення загону розміру  $K$ .

Для цього відсортуємо мінйонів за спаданням сили. Обираємо перших  $i$  з них (за новим порядком сортування). Мінімальну силу серед цих  $i$  мінйонів має саме  $i$ -ий. Тепер нам необхідно вибрати  $K$  мінімальних витривалостей серед  $i$  мінйонів, перевірити, чи їх середнє арифметичне не перевищує силу  $i$ . Якщо все добре - можна утворити загін з  $K$  мінйонів, інакше переходимо до  $i + 1$ . Аби не шукати наново  $K$  мінімальних витривалостей, використаємо структуру даних, що дозволяє знаходити максимум та видаляти його. Нам підійде черга з пріоритетом, купа або ж збалансоване бінарне дерево пошуку.

**Асимптотика:**

По часу -  $O(N \log^2 N)$

По пам'яті -  $O(N)$

### Варіант 2

Припустимо, що у нас є деякий набір мінйонів і ми хочемо обрати максимальну підмножину мінйонів так, щоб середнє арифметичне їх витривалостей не перевищило деякого  $A$ . Яким чином це зробити?

Уявімо собі масив, в якому  $i$ -ий елемент представляє кількість мінйонів з витривалістю рівно  $i$ . Побудуємо над цим масивом дерево відрізків, зберігаючи при цьому в його вершинах інформацію про суму витривалостей і кількість мінйонів на цьому піддереві. За нашою лемою ми можемо застосувати до цього дерева бінарний пошук, отримавши максимальну кількість мінйонів у загоні так, щоб середнє арифметичне їх витривалостей була не більша за  $A$ . Щоправда, часова складність бінарного пошуку з використанням дерева відрізків буде  $O(\log^2 N)$ , що не покращує варіант 1. Але вихід є, і називається він каскадним спуском по дереву. Детальніше можна дізнатися тут:

[http://en.wikipedia.org/wiki/Fractional\\_cascading](http://en.wikipedia.org/wiki/Fractional_cascading)<sup>1</sup>

Фінальний розв'язок виглядає так: ми сортуємо мінйонів за спаданням сили, розглядаємо перших  $i$ , додаємо їх витривалості до дерева відрізків. Тепер за допомогою каскадного спуску знаходимо розмір максимального загону, середнє арифметичне витривалостей якого не перевищує силу  $i$ -го мінйона.

Щодо дерева відрізків: вхідні дані не дозволяли у явному вигляді будувати дерево відрізків, оскільки витривалості могли бути завеликими, тож передбачалося, що учасник реалізує або неявне дерево відрізків (<http://bit.ly/ImplicitSegTree>) або стиснення координат (<http://bit.ly/CoordCompression>).

**Асимптотика:**

Тут  $\alpha$  - різниця між найбільшою витривалістю і найменшою витривалістю мінйонів.

По часу -  $O(N \log N)$  (для варіанту з неявним деревом відрізків  $O(N(\log N + \log \alpha))$ )

По пам'яті -  $O(N)$  (для варіанту з неявним деревом відрізків  $O(N \log \alpha)$ ).

<sup>1</sup>Тут і далі більшість статей буде англійською

## Ідея розв'язку задачі С

Запропонував і підготував: Асландуков М.

Спершу помітимо, що восьминог по своїй суті є зв'язним графом з  $K$  вершин і  $K$  ребер. Іншими словами, ми до деякого графа-дерева додали ще одне ребро, утворивши рівно один простий цикл. З цього легко знайти критерій існування восьминога: необхідно, аби сума степенів вершин була рівн  $2N$ , а також існувало хоча б три вершини зі степенями щонайменше 2.

Таким чином, варіант без максимізації кількості восьминогів виконується зовсім просто: ми об'єднуємо в цикл усі вершини зі степенем більшим за 2, а «листки» (зі степенем рівно 1) напряму під'єднуємо до тіла нашого восьминога (звісно, слідкуючи, куди слід прикріпляти скільки листків, щоб набарти потрібний степінь для кожної вершини).

Для вирішення задачі з максимізацією восьминогів помітимо, що завжди можна розбити восьминога з тілом довжини  $L (6 \leq L)$  на двох восьминогів розмірами 3 та  $L - 3$ . Таким чином, можна жадібно розбити усі вершини зі степенем не меншим за 2 на групки по 3, 4 та 5 вершин (звісно, максимізувавши їх кількість, тобто максимально групуючи по 3 вершини) і відповідним чином під'єднати вершини зі степенем 1.

### Підзадачі 1, 2

У цих підзадачах можна було помітити, що вершини степеня 3 обов'язково належать тілу восьминога, а вершини степеня 1 - «коротенькі» щупальці. Таким чином, достатньо було перевірити, що кількість вершин степеня 1 рівна кількості вершин степеня 3, а вершин степеня 3 щонайменше 3. Якщо це так, об'єднуватимемо вершини степеня 3 у цикли, а вершини степеня 1 під'єднуватимемо по одній до вершин з циклу.

### Асимптотика:

По часу -  $O(N)$

По пам'яті -  $O(N)$

## Ідея розв'язку задачі D

Запропонував: Мисак Д.

Підготували: Баренблат І., Мінаков С.

Покажемо, що задачу можна розв'язати за час  $O(N \log N + M)$ . Домовимося про таке:

- якщо  $K$  — деяка гілка, то  $K'$  позначатиме гілку перед нею;
- якщо  $K$  — деяка гілка, а  $T \leq N$ , то через  $T(K)$  позначимо набір із  $T$  гілок: саму гілку  $K$  та  $T - 1$  гілку перед нею.

Для загальності міркувань досадимо на кожну гілку без пташок по одній фіктивній пташці, що є важчою за всіх існуючих. Фіктивні пташки не зможуть впливати на рух справжніх, а самі перестрибуватимуть лише тоді, коли на їхній поточній гілці не буде жодної справжньої пташки.

**Лема 1.** Починаючи з  $N$ -го перестрибування (включно)  $N$  найлегших серед усіх пташок почнуть циклічно рухатися по колу.

**Доведення.** Без втрати загальності вважаємо, що кількість пташок дорівнює  $N$ : пташки, що не входять до числа  $N$  найлегших, не впливатимуть на їхній рух. Розглянемо тоді розташування пташок перед  $N$ -м перестрибуванням. Якби твердження леми не справджувалося, то існувала б деяка гілка  $K$ , на якій не сидить жодна пташка. Тоді перед  $(N - 1)$ -м перестрибуванням жодна пташка не сиділа на гілці  $K'$ , перед  $(N - 2)$ -м перестрибуванням пташок не було на гілці перед  $K'$  і т. д.; перед першим перестрибуванням жодної пташки не було на гілці, наступній за  $K$ . Таким чином, на кожній фіксованій з  $N$  гілок у якийсь момент часу не було пташок. Але тоді в жоден наступний момент часу кількість пташок на цій гілці не може перевищувати 1, що означало б, що загальна кількість пташок перед  $N$ -м перестрибуванням (враховуючи щонайменше одну порожню гілку) є меншою за  $N$ . Одержали суперечність.

**Наслідок 1.** Якщо  $T \geq N$ , відповідь до задачі не зміниться, якщо замість  $M$  пташок початково розглядати лише  $\min\{N, M\}$  найлегших пташок.

**Лема 2.** Якщо  $K$  — деяка гілка і  $T \leq N$ , то пташка, що на  $T$ -му перестрибуванні стрибне з гілки  $K$  на наступну, є найлегшою серед усіх пташок, що в початковий момент часу сиділи на гілках  $T(K)$  і ще не перестрибували з гілки  $K$ .

**Доведення.** Застосуємо метод математичної індукції. Для  $T = 1$  твердження очевидне. Для  $T > 1$  скористаємося твердженням леми для перестрибування  $T - 1$  та гілки  $K'$ . Тоді перед  $T$ -м перестрибуванням на гілці  $K$  буде сидіти найлегша пташок з числа тих, які ще залишилися на гілках  $T(K)$ . Вона й перестрибне з гілки  $K$  на наступному кроці.

**Лема 3.** Якщо  $K$  — деяка гілка і  $T \leq N$ , то пташка, що на  $T$ -му перестрибуванні стрибне з гілки  $K$  на наступну, є однією з найлегших  $T$  пташок серед усіх пташок, що в початковий момент часу сиділи на гілках  $T(K)$ .

**Доведення.** Оскільки за попередні  $T - 1$  перестрибування з гілки  $K$  перестрибнула не більше ніж  $T - 1$  пташка з числа найлегших  $T$ , щонайменше одна з них мала залишитися. Тоді твердження випливає з леми 2. **Лема 4.** Якщо  $T \leq N$ , то множина пташок на довільній гілці  $K$  перед  $T$ -м перестрибуванням (а отже —  $i$  вага пташки, що перестрибуватиме в момент  $T$ ) залежить лише від початкового розташування пташок на гілках  $T(K)$ . **Доведення.** Для  $T = 1$  твердження очевидне. Для  $T > 1$  двічі скористаємося твердженням леми: для моменту часу  $T - 1$  та гілок  $K$  і  $K'$ . Оскільки вміст цих гілок у момент часу  $T - 1$  однозначно визначає вміст гілки  $K$  у момент часу  $T$ , індукційний перехід доведено.

**Наслідок 2.** Якщо  $T \leq N$ , відповідь до задачі не зміниться, якщо замість  $M$  пташок початково розглядати лише  $T$  найлегших пташок з числа тих, що сидять на гілках  $T(K)$

(якщо їх було більше ніж  $T$ ). Виділити (не більше ніж)  $N$  найлегших пташок із числа (не більше ніж)  $M$  можна за лінійний від  $M$  час за допомогою ефективного пошуку порядкової статистики. Безпосередньо відповідь знайдемо тепер таким чином:

- Якщо  $T \leq N$ , просимулюємо твердження лема 2 за допомогою черги з пріоритетами: щораз додаватимемо до черги нову гілку з пташками та діставатимемо звідти вагу найлегшої пташки. Вийнята з черги на  $T$ -му кроці вага й буде шуканою.
- Якщо  $T > N$ , то підмінимо дерево з  $N$  гілок на дерево з  $2N$  гілок, що містить дві «копії» початкового дерева, які циклічно переходять одна в одну (при цьому ваги птахів тепер повторюються, але це не критично, адже те, що вони різні, у розв'язку ми не використовували). Неважко зрозуміти, що і в будь-який майбутній момент часу після перельотів пташок за встановленими правилами дерево з  $2N$  гілок залишатиметься ідентичною подвійною копією початкового дерева станом на той самий момент. Скористаємося цим: знайдемо відповідь для однієї з двох гілок нового дерева, що відповідають початковій гілці  $K$ , у момент часу  $N + T \% N$  (де  $T \% N$  позначає остачу від ділення  $T$  на  $N$ ). Згідно з лемою 1 отримана відповідь буде шуканою.

**Асимптотика:**

По часу -  $O(N \log N + M)$

По пам'яті -  $O(M)$

## Зміст

Ідея розв'язку задачі А	2
Ідея розв'язку задачі В	4
Ідея розв'язку задачі С	5
Ідея розв'язку задачі D	6

## Ідея розв'язку задачі А

Запропонував і підготував: Валлас О.

Спочатку зазначимо, що для рейтингу з точністю  $k$  знаків після десяткової коми оптимальна відповідь завжди  $\leq 10^k$ . Розглянемо це на прикладі. Нехай ми маємо рейтинг  $3.xxx$ , де  $xxx$  – 3 будь-які цифри. Припустимо, що таксист отримав  $10^3 = 1000$  оцінок 3. Тоді його рейтинг буде дорівнювати  $\frac{3 \cdot 1000}{1000} = 3.000$ , тобто рівно 3. Тепер спробуємо замінити одну оцінку 3 на 4. Тоді його рейтинг дорівнюватиме  $\frac{3 \cdot 999 + 4 \cdot 1}{999 + 1} = \frac{3001}{1000}$ . Тобто замінивши одну трійку на четвірку ми збільшили рейтинг на  $\frac{1}{1000}$ . Таким чином якщо поступово змінювати усі 3 на 4 ми зможемо отримати будь-який рейтинг від 3.000 до 4.000 з трьома знаками після коми, а отже і заданий рейтинг  $3.xxx$ .

### Підзадача 1:

Виходячи з вищезазначеного відповідь  $\leq 10$ . Тоді її можна отримати просто перебравши усі можливі комбінації оцінок 5-ма вкладеними циклами.

Складність:  $O(10^{5k})$ , де  $k$  – кількість знаків після десяткової коми.

### Підзадача 2:

Доведемо, що оптимальну відповідь завжди можна отримати тільки за допомогою оцінок двох сусідніх типів  $a$  та  $(a+1)$ , або одного типу (якщо рейтинг цілий). Припустимо, що існує оптимальна відповідь, у якій присутні НЕ сусідні оцінки  $x$  та  $y$ ,  $x < y$ . У випадку, якщо  $y - x = 3$ , ми можемо замінити їх на та  $x + 1$  та  $y - 1$ . Якщо  $y - x = 2$  або  $y - x = 4$ , можна замінити їх на дві оцінки по  $x + 1$  або  $x + 2$  відповідно. Ці дії жодним чином не впливають ні на суму, ні на кількість оцінок, а тому не впливають і на середнє арифметичне. Таким чином, будь-який набір оцінок можна звести до набору оцінок з щонайбільше двох типів. Очевидно, що ці два типи – рейтинг округлений вниз та вгору.

Тоді розв'язання полягає у тому щоб перебрати кількість оцінок першого типу, поррахувати скільки потрібно оцінок другого типу, аби отримати заданий рейтинг і перевірити, чи кількість оцінок другого типу ціла (тобто такий розподіл оцінок можливий). Нехай рейтинг рівний  $R$ , покладемо  $a = \lfloor R \rfloor$ , кількість оцінок типу  $a$  дорівнює  $x$ , а кількість оцінок типу  $a + 1$  (якщо вони необхідні) дорівнює  $y$ . Тоді, виходячи з визначення середнього арифметичного, маємо наступну рівність:  $R = \frac{x \cdot a + y \cdot (a+1)}{x+y}$ . Тоді  $x \cdot a + y \cdot (a+1) = R \cdot x + R \cdot y$ . З цього, щоб отримати точну відповідь, потрібно виконувати операції у цілих числах або у 64-бітних числах з плаваючою комою (`long double` у C++).

Складність:  $O(10^k)$ , де  $k$  – кількість знаків після коми.

### Підзадача 3:

Представимо рейтинг  $R$  у вигляді звичайного дроби, домноживши чисельник та знаменник на  $10^{18}$ . Виходячи з визначення середнього арифметичного, знаменник цього дроби  $10^{18}$  – кількість отриманих оцінок, а чисельник  $(R \cdot 10^{18})$  – їх сума. Якщо цей дріб можна скоротити, це зменшить кількість оцінок, тому що знаменник (тобто кількість оцінок) зменшиться, а відношення чисельника до знаменника (тобто рейтинг) не зміниться. Отже, чисельник і знаменник потрібно поділити на їх НСД. Припустимо, що після скорочення ми отримали дріб  $\frac{a}{b}$ . Очевидно, що отримане у знаменнику число ( $b$ ) і є оптимальною відповіддю, адже дріб нескоротний і зменшити знаменник не вдасться. Тепер потрібно довести, що ця відповідь завжди досяжна, тобто існує набір з  $b$  оцінок від 1 до 5, сума яких дорівнює  $a$ .

Доведення цього факту аналогічне доведенню у підзадачі 2. Зауважимо, що завжди  $b \leq a \leq 5b$ , бо рейтинг за умовою від 1 до 5. Тепер припустимо, що усі оцінки дорівнюють 1. Тоді чисельник дорівнюватиме  $1 * b$ , тобто  $b$ . Тепер змінимо будь-яку оцінку ( $m \neq 5$ ) на  $m + 1$ . Тоді чисельник збільшиться на 1, а знаменник залишиться рівним  $b$ . Поступово здійснюючи такі заміни, можемо примусити чисельник «пройти» через усі значення від  $b$  до  $5b$ , зокрема і через  $a$ , яке нам потрібно отримати.



Аби відновити самі оцінки, згадаємо доведений у другій підзадачі факт про те, що оптимальну відповідь завжди можна отримати з оцінок  $\lfloor R \rfloor$  та  $\lceil R \rceil$ . Якщо б усі оцінки дорівнювали  $\lfloor R \rfloor$ , чисельник був би  $b \cdot \lfloor R \rfloor$ . Але чисельник дорівнює  $a \geq b \cdot \lfloor R \rfloor$ , тому кількість оцінок  $\lceil R \rceil$  як раз дорівнює залишку, тобто  $a - b \cdot \lfloor R \rfloor$ . Решта оцінок – оцінки  $\lfloor R \rfloor$ . Також зауважимо, що через недостатню точність зберігання чисел з плаваючою комою у пам'яті комп'ютера, необхідно зчитувати рейтинг у вигляді рядка і конвертувати одразу у ціле число. Складність:  $O(k)$ , де  $k$  – кількість знаків після коми.

**Асимптотика:**

Якщо  $k$  - кількість цифр у записі оцінки, то маємо наступну складність:

По часу -  $O(k)$

По пам'яті -  $O(k)$

## Ідея розв'язку задачі В

Запропонував і підготував: Мисак Д.

Назвемо будь-яку послідовність важливих перехресть таку, що кожне наступне перехрестя в ній є водночас правішим і нижчим від попереднього, важкодоступною. Доведемо, що відповідь дорівнює розміру  $S$  максимальної важкодоступної послідовності перехресть. Зрозуміло, що меншою кількістю маршрутів ми обійтися не зможемо, адже жоден маршрут не може покрити більше, ніж одне перехрестя з числа  $S$  важкодоступних. Те, що  $S$  маршрутів має вистачити, покажемо за допомогою методу математичної індукції. Для випадку  $S = 0$  твердження очевидне. Нехай тепер воно виконане для  $S - 1$ . Перший з  $S$  маршрутів пустимо таким чином:

1. Робимо декілька (на першому кроці, можливо, нуль) кроків угору, поки не прийдемо в рядок, у якому залишаються важливі перехрестя (або у найвищий рядок).
2. Далі робимо декілька (можливо, нуль) кроків праворуч, поки не дійдемо до найправішого важливого перехрестя в поточному рядку (або, якщо це верхній рядок, до його кінця).
3. Якщо ми ще не в кінцевому пункті, повертаємося до кроку 1).

Подібна побудова маршруту гарантує, що а) правіше чи нижче від лінії маршруту не буде жодного важливого перехрестя і б) кожне перехрестя, де автобус повертав ліворуч, є важливим. Розглянемо довільну важкодоступну послідовність для нової (зменшеної) множини важливих перехресть. Якби вона мала розмір  $S$ , ми могли б від найправішого/найнижчого перехрестя цієї послідовності опустити пряму лінію до перетину з проведеним маршрутом, узяти найправіше важливе перехрестя в тому рядку на маршруті, де лінія його перетнула, і додати це перехрестя до важкодоступної послідовності з  $S$  вершин. Це дало б нам важкодоступну послідовність із  $S + 1$  вершини для початкової множини важливих перехресть, що суперечить зробленому вибору максимальної важкодоступної послідовності. Отже, нова максимальна важкодоступна послідовність має довжину, не більшу за  $S - 1$ , і за припущенням індукції, крім одного вже проведеного маршруту, нам достатньо пустити ще  $S - 1$  маршрут, щоб покрити всі важливі перехрестя. Доведено. Тепер безпосередньо підрахуємо розмір максимальної важкодоступної послідовності важливих перехресть. Це можна зробити за допомогою одного обходу таблиці зліва направо зверху вниз: якщо домовитися, що  $f(i, j)$  дорівнює відповіді для прямокутника, лівим верхнім кутом якого є ліве верхнє перехрестя, а правим нижнім — перехрестя на перетині  $i$ -го рядка та  $j$ -го стовпця ( $1 \leq i \leq N, 1 \leq j \leq M$ ), то матимемо такі рекурентні співвідношення:

$$f(i, j) = \begin{cases} 0, & \text{якщо } i = 0 \text{ або } j = 0 \\ f(i - 1, j - 1) + 1, & \text{якщо перехрестя } (i, j) \text{ важливе} \\ \max\{f(i - 1, j), f(i, j - 1)\}, & \text{якщо перехрестя } (i, j) \text{ не є важливим} \end{cases} \quad (1)$$

Правильність формули для останнього випадку випливає з того, що перехрестя у правому стовпці та перехрестя у нижньому рядку не можуть водночас належати одній і тій самій важкодоступній послідовності. Отже, залишається вивести значення  $f(N, M)$ . Складність побудованого алгоритму —  $O(N \cdot M)$ . Альтернативним підходом є послідовно один за одним жадібно будувати маршрути, користуючись кроками 1) — 3) із доведення вище, поки не буде покрито всі важливі перехрестя. Це також можна зробити за  $O(N \cdot M)$  часу, якщо ввести допоміжну індексацію по рядках чи стовпцях.

**Асимптотика:**

По часу -  $O(N \cdot M)$

По часу -  $O(N \cdot M)$

## Ідея розв'язку задачі С

Запропонував і підготував: Мисак Д.

У **підзадачі 1** відповідь на кожен запит — кількість доданих на даний момент слів.

У **підзадачі 2** результат запиту — одиниця, якщо сумарна довжина всіх доданих на даний момент слів, складена з їхньою кількістю, не більша за  $L + 1$ , а в іншому разі — двійка.

**Підзадачах 3 й 4**, у яких слова завжди додаються в кінець тексту, можна розв'язати за лінійний від кількості запитів час, опрацювуючи всі запити по черзі та пам'ятаючи поточні кількість рядків і кількість вільних позицій в останньому рядку тексту.

**Підзадачах 5 та 6**, у яких слова завжди додаються на початок тексту, також можна розв'язати за лінійний від кількості запитів час, зберігаючи в масиві кількість рядків, утворених після додавання кожного слова, а також підтримуючи в окремих змінних поточні кількість слів і кількість вільних позицій у першому рядку тексту. Після додавання нового слова або кількості слів у першому рядку збільшується на 1, а кількість вільних позицій відповідним чином зменшується, або одне чи кілька слів з кінця рядка переходять у наступний; тоді нова загальна кількість рядків — це кількість рядків, збережена для теперішнього першого слова другого рядка, збільшена на 1.

Альтернативний спосіб — зауважити, що якби слова додавалися завжди не на початок, а в кінець тексту, то загальна кількість рядків у кожен момент часу залишалася б такою самою (а потім скористатися тим самим алгоритмом, що й для підзадач 3 й 4). Кількість не могла зменшитися, бо інакше ми б розгорнули текст на 180 градусів, зсунули слова на вільні позиції ліворуч (якщо такі є) і отримали б початковий текст, що вмістився у меншу кількість рядків, ніж він вмістився насправді. З симетричних міркувань кількість рядків не могла й збільшитися.

**Підзадачах 7 і 8** так само можна розв'язати за лінійний час. Для цього застосуємо перший підхід, описаний для **підзадач 5 і 6**, додатково зберігаючи для кожного слова кількість вільних позицій в останньому рядку тексту, якщо першим словом тексту є дане. Тепер розіб'ємо текст на дві частини — множину слів, доданих на початок, та множину слів, доданих у кінець. Другу частину тексту розглядатимемо як розгорнутий на 180 градусів фрагмент, у який слова так само додавали на початок. Якщо тепер останні рядки двох частин (тобто насправді нижній рядок першої частини та верхній рядок другої частини) вміщуються — разом із пробілом між ними — в одному рядку, то загальна кількість рядків у тексті є сумою кількостей для двох частин, зменшеною на 1; інакше це просто сума кількостей рядків у двох частинах тексту. Коректність такого твердження впливає з міркувань, цілком аналогічних доведенню правильності альтернативного розв'язку для **підзадач 5 і 6**. Під час складання двох кількостей треба акуратно розглядати випадки, коли одна з двох частин тексту порожня.

**Асимптотика:**

По часу -  $O(N)$

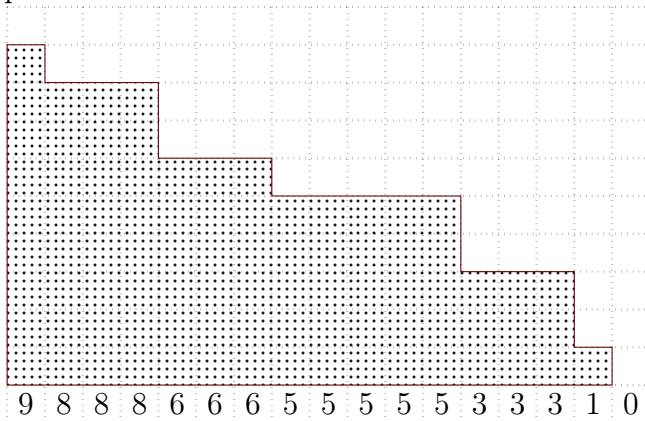
По пам'яті -  $O(N)$

## Ідея розв'язку задачі D

Запропонував: Міхно М..

Підготували: Асландуков М., Баренблат І.,  
Міхно М., Селіванов А., Стречень М., Фекете І.

Спершу розглянемо випадок **одного кута**. Нехай після певної кількості кроків маємо деяку заливку. Помітимо, що, по-перше, можемо представити цю картинку масивом, у якому  $i$ -ий елемент буде представляти висоту  $i$ -го стовпчика, а по-друге, цей масив буде незростаючим.



Побудуємо над нашим масивом дерево відрізків. Навіщо? Справа в тому, що для додавання прямокутника нам знадобиться вміння швидко присвоювати на відрізку, а для операції знаходження кількості зафарбованих клітинок - вміння дізнаватися суму на відрізку. Крім того, пізніше нам знадобиться інформація про найлівіший і найправіший елемент відрізка (маються на увазі саме ті відрізки, що зберігаються в дереві відрізків, а не будь-які).

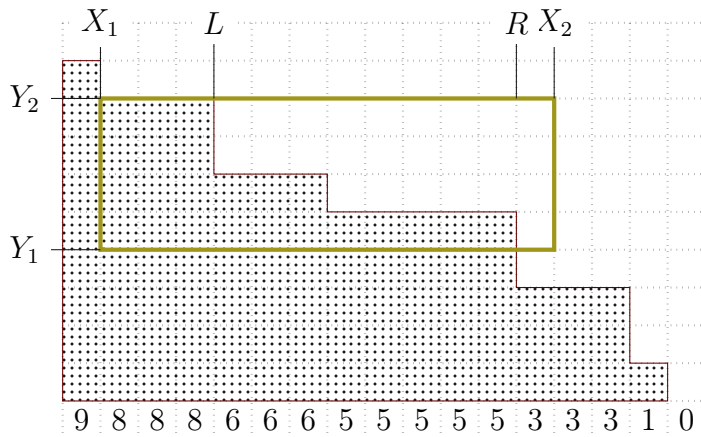
Важливою підзадачею є пошук найлівішого елемента масиву, що **не менший** за деяке  $X$  (звісно, у відсортованому масиві). Це можна було б робити бінарним пошуком, але через звернення до дерева відрізків матимемо  $O(\log^2 N)$  на запит. Для покращення асимптотики до  $O(\log N)$  застосуємо метод каскадного спуску. Нехай ми шукаємо кандидата серед чисел у підвідрізку  $[L..R]$ . Покладемо  $mid = \lfloor \frac{L+R}{2} \rfloor$ . Якщо найлівіший елемент відрізка  $[mid + 1..R]$  (тобто  $(mid + 1)$ -ий) менший за  $X$ , то усі цього підвідрізка також менші за  $X$ , тому слід розглядати підвідрізок  $[L..mid]$ . Інакше розглядаємо  $[mid + 1..R]$ .

### Приклад реалізації пошуку методом каскадного спуску

```

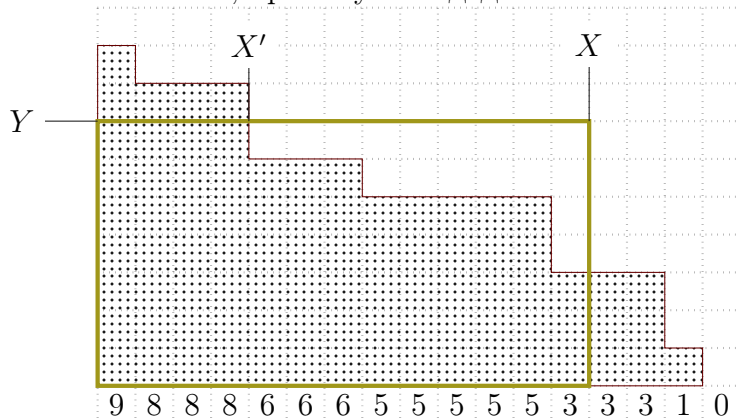
1  int last_not_less(int val){
2      int L = 1, R = size, _i = 1; // L, R - межі відрізка, який ми розглядаємо
3      push(1, size);                // _i - номер цього відрізка в дереві відрізків
4
5      if(mostleft[1] < val)          // Якщо перший елемент менший за val, то вважатимемо
6          return 0;                  // потрібним фіктивний нульовий елемент
7      while(L < R){
8          push(_i*2, (R - L + 1)/2);
9          push(_i*2 + 1, (R - L + 1)/2);
10         if(mostleft[_i*2 + 1] < val)
11             R = (L + R)/2,
12             _i = _i*2;
13         else
14             L = (L + R)/2 + 1,
15             _i = _i*2 + 1;
16     }
17     return R;
18 }
```

Тепер навчимося дізнаватися відповідь для певного прямокутника. Розіб'ємо наші стовпчики на три групи - ті, що повністю покривають наш прямокутник (на ілюстрації стовпчики від 2 до 4), ті, що частково покривають наш прямокутник (на іл. від 6 до 12) та ті, що жодним чином прямокутник не зачіпають (на іл. 13-ий стовпчик). Межі цих груп можна шукати за допомогою тієї самої «важливої підзадачі».



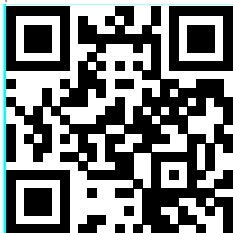
Легко бачити, що для першої групи тестів площа зафарбованих клітинок буде складати  $(Y_2 - Y_1) * (X_2 - X_1)$ , для третьої 0, а для другої  $get(L, R) - Y_1 * (R - L)$  ( $get(L, R)$  - сума на підвідрізку  $[L..R]$ , яку нам швидко знаходить дерево відрізків).

Навчимося додавати прямокутник. Нам треба знайти перший стовпчик, що лежить нижче ніж наш  $Y$  (точка  $X'$ ). Тепер просто присвоїмо усім елементам на відрізку  $[X'..X]$  значення  $Y$ . Готово, прямокутник додано.



#### Перехід до 4 кутів.

Перше, що можемо помітити - якщо прямокутники, що зберігаються у деревах відрізків, не перетинаються, ми можемо легко отримувати відповідь про кількість зафарбованих клітинок - просто незалежно дізнаємося відповідь для кожного з кутків і просумуємо. Як підтримувати стан без перетинів, пояснено в презентації за наступним посиланням (з великою кількістю зображень): <http://bit.ly/uoi2018-2-D>. Матеріали олімпіади можна буде знайти за посиланням <http://bit.ly/UOI-2018>. Наступні QR-коди ведуть туди ж:



#### Асимптотика:

По часу -  $O(Q \log N)$

По пам'яті -  $O(N)$  (щоправда з великою константою).