Зміст

Ідея розв'язку задачі A

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 ≤ 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, порахувати 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 плавоючою комою (`long double` у C++).

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

Підзадача 3:

Представимо рейтинг R у вигляді звичайного дробу, домноживши чисельник та знаменник на 10^{18}. Виходячи з визначення середнього арифметичного, знаменник цього дробу 10^{18} – кількість отриманих оцінок, а чисельник $(R \cdot 10^{18})$ – їх сума. Якщо цей дріб можно 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 $\frac{a}{b}$. Очивидно, що отримане у знаменнику число (b) і є оптимальною відповіддю, адже дріб нескоротний і зменшити знаменник не вдасться. Теперь потрібно довести, що ця відповідь завжди досяжна, тобто існує набір з b оцінок від 1 до 5, сумма 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. Теперь припустимо, що усі оцінки дорівнюють 1. Тоді чисельник дорівнюватиме $1 * b$, тобто b. Тепер змінемо 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 B

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 ($1 \le i \le N, 1 \le j \le M$), то матимемо такі рекурентні співвідношення:

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

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 C

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, опрацьовуючи 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 кількість 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.

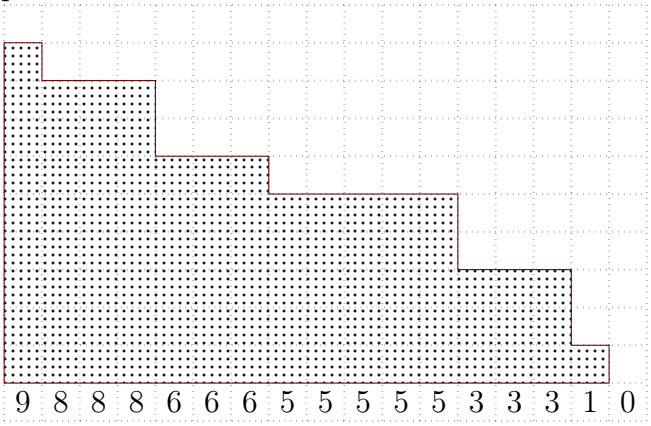
Підзадачах 7 і 8 так само можна розв'язати за лінійний час. Для цього застосуємо перший підхід, описаний для **підзадач 5 і 6**, додатково зберігаючи для кожного слова кількість вільних позицій в останньому рядку тексту, якщо першим словом тексту є дане. Тепер розіб'ємо текст на дві частини — множину 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) вміщаються — 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 випливає 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.



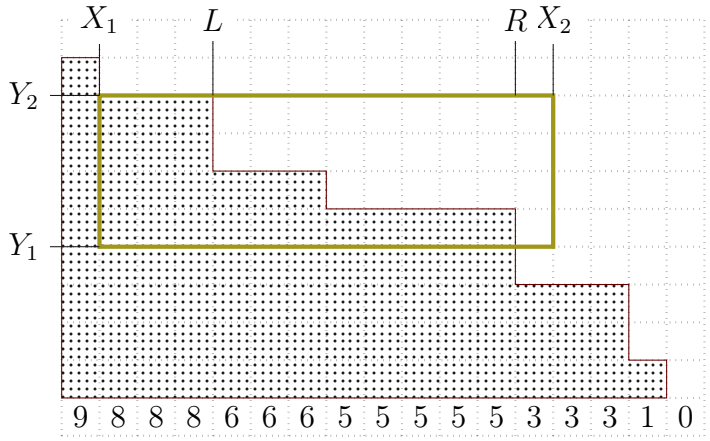
Побудуємо над нашим масивом дерево відрізків. Навіщо? Справа в тому, що для додавання прямокутника нам знадобиться уміння швидко присвоювати на відрізку, а для операції знаходження кількості зафарбованих клітинок - уміння дізнаватися суму на відрізку. Крім того, пізніше нам знадобиться інформація про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

Важливою підзадачею є пошук найлівішого елементу 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 відрізку $[mid+1..R]$ (тобто $(mid+1)$-ий) менший за X, то усі цього підвідрізку також менші за X, тому слід розглядати підвідрізок $[L..mid]$. Інакше розглядаємо $[mid+1..R]$.

Приклад реалізації пошуку методом каскадного спуску

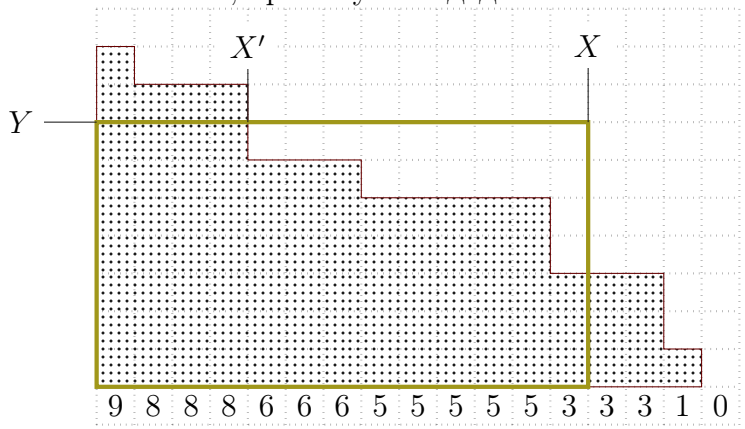
```
int last_not_less(int val){
    int L = 1, R = size, _i = 1;//  L, R - межі відрізку, який ми розглядаємо
    push(1, size);              //  _i - номер цього відрізку в дереві відрізків

    if(mostleft[1] < val)       //  Якщо перший елемент менший за val, то вважатимемо
        return 0;               //  потрібним фіктивний нульовий елемент
    while(L < R){
        push(_i*2, (R - L + 1)/2);
        push(_i*2 + 1, (R - L + 1)/2);
        if(mostleft[_i*2 + 1] < val)
            R = (L + R)/2,
            _i = _i*2;
        else
            L = (L + R)/2 + 1,
            _i = _i*2 + 1;
    }
    return R;
}
```

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 $get(L, R) - Y_1 \cdot (R - L)$ ($get(L, R)$ - 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).