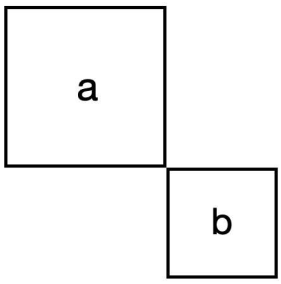
Задача A. Парижский полумарафон

Автор задачи: Демид Кучеренко, разработчик: Александр Волков

Правильный ответ на задачу — это сумма двух максимумов. Давайте рассмотрим два квадрата с наибольшей стороной (пусть они будут равны a и b ($b \leqslant a$)). Заметим, что их можно поместить только в квадрат со стороной хотя бы $a + b$. Если расположить их следующим образом, то меньшие квадраты можно расположить между ними, вписав их в тот же квадрат со стороной $a + b$.



Задача B. Трамвай

Автор и разработчик задачи: Глеб Костылев

Формально, требуется найти такую прямую, что суммарное расстояние от неё до всех точек из набора минимально. Назовём это прямую оптимальной.

Покажем, что есть такая оптимальная прямая, что хотя бы одна точка из набора лежит на ней. Допустим, такой не существует, тогда рассмотрим оптимальную прямую, на которой не лежит ни одной точки. Хотя бы в одной полуплоскости от нее лежит m точек, где $m \geqslant \lceil \frac{n}{2} \rceil$. Сдвинем прямую в сторону этой полуплоскости, до тех пор, пока на ней не будет лежать хотя бы одна точка. Тогда для m точек ответ улучшится суммарно на lm, для остальных $(n - m)$ — ухудшится на $l(n - m)$, где l — расстояние, на которое мы сдвинули прямую. Так как $m \geqslant n - m$, ответ не ухудшится.

Докажем, что если прямая проходит ровно через одну точку, то из неё можно получить прямую с меньшим суммарным расстоянием до точек, проходящую сразу через две точки из набора. Пусть координаты точки из набора, через которую проходит прямая, (x_0, y_0), и её полярный угол φ. Рассмотрим произвольную точку (x_i, y_i) из набора, не совпадающую с (x_0, y_0). Пусть (X_i, Y_i) — вектор от (x_0, y_0) до (x_i, y_i), то есть $(x_i - x_0, y_i - y_0)$. Если модуль (X_i, Y_i) равен d_i, то $X_i = d_i \sin \alpha_i$, $Y_i = d_i \cos \alpha_i$, где α_i — полярный угол (X_i, Y_i), и тогда расстояние от i-й точки то прямой равно $d_i |\sin(\varphi - \alpha_i)|$. Тогда расстояние от всех точек до прямой — это $\sum_{i=1}^{n-1} d_i |\sin(\varphi - \alpha_i)|$. Функция $|\sin x|$ строго выпуклая на каждом отрезке вида $[\pi k; \pi(k+1)]$, $k \in \mathbb{Z}$, то есть для любых двух точек x_1, x_2 с одного такого отрезка $\frac{c_1|\sin x_1| + c_2|\sin x_2|}{c_1 + c_2} < |\sin \frac{c_1 x_1 + c_2 x_2}{c_1 + c_2}|$ при $c_1, c_2 > 0$ (факт математического анализа). Из выпуклости следует, что для любых $x_1 < x_2 < x_3$ с одного отрезка $|\sin x_1| < |\sin x_2|$ или $|\sin x_3| < |\sin x_2|$. Тогда $d_i|x - \alpha_i|$ строго выпуклая на отрезках вида $[\alpha_i + \pi k; \alpha_i + \pi(k+1)]$. Для каждого i рассмотрим отрезок данного вида, который покрывает φ, пусть это отрезок $[l_i; r_i]$. Так как $\varphi \neq \alpha_i$, то $l_i < \varphi < r_i$. Значит, у этих отрезков есть непустое пересечение $[L; R]$, $L = l_j$, $R = r_k$, на котором $\sum_{i=1}^{n-1} d_i |\sin(x - \alpha_i)|$ строго выпуклая как сумма строго выпуклых функций. Тогда $\sum_{i=1}^{n-1} d_i |\sin(L - \alpha_i)| < \sum_{i=1}^{n-1} d_i |\sin(\varphi - \alpha_i)|$ или $\sum_{i=1}^{n-1} d_i |\sin(R - \alpha_i)| < \sum_{i=1}^{n-1} d_i |\sin(\varphi - \alpha_i)|$. Значит, одна из прямых, проходящих через (x_0, y_0) и имеющих полярный угол α_j или α_k, будет иметь меньшее суммарное расстояние до всех точек, чем прямая с полярным углом φ, которую мы взяли изначально.

Найдем из всех прямых, проходящих через (x_0, y_0), оптимальную. Отсортируем (X_i, Y_i) по возрастанию полярному углу и будем перебирать их в порядке сортировки. Пусть мы рассматриваем

вектор (X, Y), и его модуль равен d. Суммарное расстояние от остальных точек до прямой, заданной им, — это $\sum_{i=1}^{n-1} \frac{|XY_i - YX_i|}{d}$ по формуле ориентированной площади параллелограмма, натянутого на 2 вектора (или формуле «векторного произведения»). Для всех (X_i, Y_i), расположенных по направлению против часовой стрелки от (X, Y), модуль раскроется как $XY_i - YX_i$, а для расположенных по направлению по часовой стрелке как $YX_i - XY_i$. Чтобы быстро посчитать расстояние, осталось научиться поддерживать при переборе сумму X_i и Y_i для векторов в обоих направлениях от вектора (X, Y). Так как вектора отсортированы по возрастанию полярного угла, достаточно хранить указатель на позицию в отсортированном массиве, где находился бы $(-X, -Y)$. Тогда все векторы от (X, Y) до $(-X, -Y)$ будут находиться по направлению против часовой стрелки от (X, Y), а все остальные — по часовой стрелке. Будем двигать этот указатель вместе с (X, Y) и пересчитывать суммы.

Итоговая асимптотика $O(n \log_2 n)$ для каждой (x_0, y_0), то есть $O(n^2 \log_2 n)$ суммарно.

Альтернативное решение задачи использует технику «вращающегося сканлайна». Переберём все прямые между какими-либо двумя точками в порядке возрастания их угла с осью Ox и будем поддерживать все точки из набора по возрастанию их проекций на нормаль прямой. Пусть сейчас мы рассматриваем прямую $Ax + By + C = 0$, $A^2 + B^2 = 1$. Тогда суммарное расстояние от точек до неё равно $\sum_{i=1}^{n} |Ax_i + By_i + C|$. Так как точки отсортированы по возрастанию проекции на нормаль, $\sum_{i=1}^{n} |Ax_i + By_i + C| = \sum_{i=1}^{k} -Ax_i - By_i - C + \sum_{i=k+1}^{n} Ax_i + By_i + C = -A(\sum_{i=1}^{k} x_i) - B(\sum_{i=1}^{k} y_i) - Ck + A(\sum_{i=k+1}^{n} x_i) + B(\sum_{i=k+1}^{n} y_i) + C(n-k)$ для некоторого k. Найти k можно бинарным поиском, а префиксные суммы координат — поддерживать при перестановке точек. Однако, условие не гарантирует, что на каждой прямой лежит не более двух точек из набора, поэтому данные случаи необходимо отдельно рассматривать при написании сканлайна.

Задача C. Ст. 243 УК РФ

Автор задачи: Михаил Грибакин, разработчик: Никита Зибницкий

Заметим, что `XOR` чисел a, выписанных h раз, равен 0, если h кратно 2, и a иначе. Тогда, если бы массив h не изменялся, в запросе требовалось бы вывести `XOR` всех a_i с отрезка $[l; r]$ таких, что h_i не кратно 2. Эту величину можно находить за $O(1)$, используя массив префиксных `XOR`-ов.

Как теперь решить полную задачу? Заметим, что если x кратно 2, то чётность НОД(x, h) совпадает с чётностью h. Другими словами, после изменений чётность элементов в массиве h не изменилась, то есть всё ещё работает предыдущее решение.

Если же x не кратно 2, то число НОД(x, h) нечётно при любом h, тогда достаточно вывести `XOR` всех a_i с отрезка $[l; r]$.

Поскольку операция `XOR` обратна сама себе ($a \oplus a = 0$, $a \oplus 0 = a$), можно вычислять `XOR` на отрезке аналогично префиксным суммам. Пусть нам нужно найти XOR всех чисел на отрезке $[L; R]$. Просто запишем его в виде:

$$a_L \oplus a_{L+1} \oplus \ldots \oplus a_R = (a_0 \oplus a_1 \oplus \ldots \oplus a_R) \oplus (a_0 \oplus a_1 \oplus \ldots \oplus a_{L-1})$$

Итоговая асимптотика: $O(n + q)$.

Задача D. Покупка одежды

Автор и разработчик задачи: Никита Насонков

Заметим, что для любых целых неотрицательных чисел a и b выполняется неравенство $a + b \geqslant a \oplus b$. В качестве доказательства рассмотрим побитовые представления $a + b$ и $a \oplus b$:

1. Пусть i-й бит числа a равен 0 и i-й бит числа b тоже равен 0. Тогда как в $a + b$, так и в $a \oplus b$ за счёт i-го бита добавится 0.

2. Пусть i-й бит числа a равен 0, а i-й бит числа b равен 1. Тогда как в $a + b$, так и в $a \oplus b$ за счёт i-го бита добавится 2^i.

3. Пусть i-й бит числа a равен 1, а i-й бит числа b равен 0. Тогда как в $a + b$, так и в $a \oplus b$ за счёт i-го бита добавится 2^i.

4. Пусть i-й бит числа a равен 1 и i-й бит числа b тоже равен 1. Тогда в $a + b$ за счёт i-го бита добавится 2^{i+1}, а в $a \oplus b$ добавится 0.

Таким образом в первых трёх случаях $a+b = a \oplus b$, а в последнем случае $a+b > a \oplus b$. Следовательно, в общем случае $a + b \geqslant a \oplus b$, ч и т. д.

Цену последнего оставшегося в результате действий продавца наименования можно представить в виде $s_1 + s_2 + ... + s_p$, где s_i — это побитовое исключающее «ИЛИ» какого-то непустого подмножества элементов исходного массива a (причём каждый элемент массива a принадлежит ровно одному подмножеству). Заметим, что если мы в каждом s_i заменим побитовое исключающее «ИЛИ» на сумму, то ответ не уменьшится за счёт доказанного выше неравенства. Таким образом, если мы просто просуммируем цены всех наименований одежды, то мы достигнем оптимального ответа.

Итого задача сводится к тому, чтобы посчитать сумму элементов в массиве a и вывести её (нужно только не забыть использовать 64-битный тип данных, так как ответ может не влезать в 32-битный тип). Асимптотическая сложность алгоритма $O(n)$.

Задача E. Автошкола «Драйв»

Автор задачи: Данил Лебедев, разработчики: Данил Клищ, Савелий Григорьев

Поскольку длины маршрутов всех курсантов одинаковые и равняются k, то положение всех курсантов на доске зависит только от остатка при делении текущего времени на k. Давайте построим граф, где вершинами будут тройки (x, y, t), $x \in [1, n]$, $y \in [1, n]$, $t \in [0, k-1]$. (x, y) обозначает клетку доски, а t — остаток при делении времени на k. Рёбра в данном графе будут построены в соответствии с переходами на доске:

- $(x, y, t) \to (x + 1, y, (t + 1) \mod k)$,
- $(x, y, t) \to (x - 1, y, (t + 1) \mod k)$,
- $(x, y, t) \to (x, y + 1, (t + 1) \mod k)$,
- $(x, y, t) \to (x, y - 1, (t + 1) \mod k)$,
- $(x, y, t) \to (x, y, (t + 1) \mod k)$.

Рёбра строятся с учётом того, что запрещено переходить в клетку с препятствием и запрещено сталкиваться с курсантами.

Осталось найти кратчайший маршрут из $(x_S, y_S, 0)$ до $(x_F, y_F, \_)$ (третий параметр конечной клетки любой). Это можно сделать алгоритмом обхода в ширину за асимптотику $\mathcal{O}(n^2 k)$.

Задача F. Командные сборы

Автор и разработчик задачи: Александр Степаненко

Давайте для удобства сделаем массив cnt, в котором cnt_i обозначает количество предметов ценностью i. Заметим, что т.к. у нас всего n предметов, то максимальная размерность любого множества, а также его ценность не больше n. Значить все элементы большие n не внесут вклада в множество, в котором находятся, а потому их можно не учитывать.

Теперь поймем, что делать больше чем cnt_0 множеств нет смысла, потому что во всех, кроме первых cnt_0, не будет элемента 0, а значит их ценность будет 0. Рассмотрим произвольные элементы x и y $(x < y)$. Если $cnt_y > cnt_x$, то элементы x можно будет использовать только cnt_y раз, потому что во всех остальных множествах точно не будет элементов y, а значит ценность будет точно меньше x и потому наличие этого элемента ничего не изменит. То есть теперь мы выяснили, что для всех элементов x и y $(x < y)$, мы можем использовать x **точно не больше** чем y. Тогда построим массив cnt', как массив префиксных минимумов массива cnt. Сумма всех значений cnt' и будет ответом на задачу.

Задача G. Слишком крутая задача

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Павел Ральников

Пусть a — массив крутостей шуб после оптимальной игры Кена, а x — позиция максимума в нем. Заметим, что тогда можно нажать на все кнопки, где $l \leqslant x \leqslant r$, так как если позиция минимума вне этого отрезка, то мы получаем $+1$ к ответу, а если внутри, то не ухудшаем ответ. Из этого следует, что минимум будет достигаться либо в позиции 1, либо в позиции m.

Тогда можно рассмотреть кнопки, не затрагивающие позицию 1 (аналогично сделать для m) и найти позицию, на которую действует наибольшее количество кнопок. Это можно уже сделать с помощью сканирующей прямой. Отрезок (l, r) — это два события: $(l, 0)$ — отрезок открывается и $(r, 1)$ — отрезок закрывается. Отсортируем эти события и пройдемся по ним, поддерживая количество открытых отрезков на данный момент. Максимум среди этих величин и есть то, что нам нужно. Решение работает за $O(n \log n)$.

Задача H. Семечки в Ленинском

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Савелий Григорьев

Научимся решать задачу для заданного массива. Заметим, что мы хотим сделать все элементы равными максимальному значению массива. Давайте для удобства будем решать аналогичную задачу, но ее проще реализовать. Сделаем массив $b_i = \max(a) - a_i$. Тогда вместо операции добавления мы будем делать -1 на отрезке и наша цель сделать все элементы массива равными 0.

Пусть pos — позиция минимального элемента массива b. Тогда мы можем b_pos раз применить операцию на всем массиве, а после рекурсивно решить задачу слева и справа относительно позиции pos. Но это нам не помогает решить задачу для всех сдвигов. Давайте для каждого элемента найдем ближайший меньше либо равный элемент слева (обозначим за l_i) и ближайший меньший справа (обозначим за r_i) — это можно сделать с помощью линейных алгоритмов. Тогда можно заметить, что когда мы будем рассматривать в нашем рекурсивном решении подотрезок массива $l_i + 1 \dots r_i - 1$ мы добавим к ответу величину равную $b_i - \max(b_{l_i}, b_{r_i})$. Мы нашли решение задачи за $O(n)$ для фиксированного массива a.

Если мы будем рассматривать все сдвиги массива, то ближайшие слева/справа меньшие либо не меняются, либо их просто в таком случае не существует и мы считаем, что значения в них равны 0. Соответственно, если мы зафиксируем позицию i (хотим для каждой позиции массива понять на сколько она изменяет ответ), то у нас есть три отрезка сдвигов, для которых мы можем прибавлять разные значения. Отсюда мы получаем $O(n)$ отрезков, на которых нужно добавлять некоторые значения. Так как нам необязательно делать эти добавления сразу, мы можем завести массив add_i — величина, которую нужно прибавить всем элементам, начиная с i-й позиции. Тогда добавление на отрезке $l \dots r$ числа x изменяет этот массив следующим образом: $add_l = add_l + x$, $add_{r+1} = add_{r+1} - x$. Ответ на задачу $ans_i = \sum_{j=1}^{i} add_i$. Решение этой подзадачи за $O(n)$.

Дальше нам нужно учиться считать суммарную стоимость. Подход будет аналогичен, только теперь нужно будет прибавлять более сложную функцию. Если оба ближайших элемента будут в сдвиге, то мы будем прибавлять $(r_i - l_i - 1)^2 \cdot (b_i - \max(b_{l_i}, b_{r_i})$ — фиксированное число. Если же не существует ближайшего слева/справа, то мы будем прибавлять величину вида $(r_i - s + 1)^2 \cdot (b_i - b_{r_i})$ или $(s + n - i + 1)^2 \cdot (b_i - b_{l_i})$, где s — индекс начала позиции сдвига. Оба случая решаются аналогично, если раскрыть скобки, то получится выражение вида $a \cdot s^2 + b \cdot s + c$, где a, b, c — известный коэффициенты. Давайте тогда ответ для фиксированного сдвига будем представлять в таком виде и нам нужно найти значение каждого коэффициента, а это уже просто прибавление фиксированного числа на отрезке, что мы разобрали в предыдущем абзаце. Также решается за $O(n)$.

Задача I. Ролики

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Павел Ральников

Заметим, что если a_i делится на a_k и a_j делится на a_k, то $\gcd(a_i, a_j)$ делится на a_k. Давайте для каждого g посчитаем количество пар из массива, таких что gcd равен g.

Для этого сделаем dp_g — количество пар массива с таким gcd. Для этого нам потребуется массив количеств — cnt_x количество вхождений числа x в массив. Чтобы посчитать dp_g, нужно узнать сколько чисел делится на g, а именно сумму $s = cnt_g + cnt_{2g} + \ldots + cnt_{kg}$, где $kg \leqslant n$. Соответственно пар чисел таких $\frac{s \cdot (s-1)}{2}$. Но не у всех них gcd равен g, он может быть равен числу кратному g. Поэтому $dp_g = \frac{s \cdot (s-1)}{2} - dp_{2g} - \ldots - dp_{kg}$. Эта динамика считается за $\sum\limits_{i=1}^{n} \frac{n}{i} = O(n \log n)$.

Теперь поймем какие gcd нам не подходят. Если есть число x в массиве ($cnt_x > 0$), то все g кратные x нам не подходят. Это также считается за $O(n \log n)$. Дальше осталось просуммировать все dp_g, где число g не делится ни на одно число из массива.

Задача J. Выборы в Карабаше

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Екатерина Шиляева

Чтобы число делилось на 2^i последние i разрядов битовой записи должны быть равны 0. Для удобства перевернем битовую запись числа, чтобы целью наших операций было обнуление первых i битов. Также пусть $zero$ — количество битов равных 0 в исходной строке.

Если $i > zero$, то очевидно, что ответ равен -1, так как наши операции не изменяют количество нулей и единиц в числе. Иначе ответ существует и научимся вычислять наименьшее количество операций. Рассмотрим все $j \leqslant i$, где $s_j = 1$. Эти единицы нужно убрать из нашего префикса, заменив их ближайшими нулями после позиции i. Будем идти по строке слева направо, поддерживать количество единиц на нашем префиксе (обозначим за cnt), сумму их позиций (обозначим за sum_one). Также нам нужно будет поддерживать сумму ближайших cnt позиций нулей после нашего элемента i (обозначим за sum_zero). Это можно делать с помощью указателя. Ответ равен $sum\_zero - sum\_one$. Это одновременно и оценка снизу и можно жадно показать как получить этот ответ за такое количество операций.

Для начала давайте все единицы поставим в конец нашего префикса. Самую правую единицу поставим на позицию i и так далее. Это мы сделаем за $\sum\limits_{j=i-cnt+1}^{i} (j - pos)$, где pos это позиция соответствующей единицы, то есть в общем случае на самом деле $= -sum\_one + \sum\limits_{j=i-cnt+1}^{i} j$. Теперь мы хотим на эти позиции поставить нули и будем делать это cnt раз жадно, сделав $\sum\limits_{j=i-cnt+1}^{i} (pos - j)$ операций, где pos — позиция 0. Опять же, суммарно это $= sum\_zero - \sum\limits_{j=i-cnt+1}^{i} j$. Наш ответ это сумма этих двух, что равно $sum\_zero - sum\_one$.

Задача K. Магазин METRO

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Екатерина Шиляева

Заметим, что задачу можно решать независимо для каждого товара. Пусть y ответ на задачу. Заметим, что $y - x \leqslant 18$ и задачу можно решить перебором. Докажем этот факт.

Если разница двух чисел больше 18, то точно найдется последовательность из 10 чисел, где различается только последняя цифра. Соответственно, если сумма цифр числа с последней цифрой 0 равна x, то были суммы цифр равные $x, x+1, \ldots, x+9$. Здесь ровно 10 последовательных чисел и для каждого k от 1 до 10 найдется число из этого промежутка, которое делится на k.

Задача L. Переплава через реку

Автор и разработчик задачи: Екатерина Шиляева

Давайте построим ориентированный невзвешенный граф, в котором вершинами будут точки координатной прямой, а ребро из вершины u в вершину v будет проведено тогда и только тогда, когда есть сом с координатами бревна $l_i = u$, $r_i = v$. Заметим, что любой путь от вершины x до вершины y в таком графе будет являться корректно выложенной последовательностью бревен,

покрывающих весь участок реки от x до y. Также нетрудно заметить, что в построенном графе расстояние от вершины L до вершины R будет равно ответу на изначальную задачу, притом бревна, отвечающие за ребра на кратчайшем пути, будут теми, которые Барби будет выгодно выбрать. Еще заметим, что в нашем графе нас интересуют только те вершины, которые встречаются среди точек L, R и всех l_i, r_i, ведь из и в остальные вершины не будет проведено ребер, и они никак не повлияют на кратчайший путь из L в R.

Теперь решение задачи будет выглядеть следующим образом: заведем *map*, в котором будем хранить ребра графа списками смежности, и дополнительно запомним номер сома, представляющего бревно для каждого ребра (для восстановления ответа). Теперь все, что осталось сделать — найти кратчайший путь из L в R, что можно сделать с помощью BFS. Вместо использования *map* можно воспользоваться алгоритмом сжатия координат и хранить граф и расстояния в массиве.

Асимптотика решения $O(n \log n)$ при использовании *map* или сжатия координат, или $O(n)$ при использовании *hash map*.

Задача M. Подотрезки кратные k

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Демид Кучеренко

Заметим, что если $\prod_{t=i}^{j} a_t$ делится на k, то и $\prod_{t=i}^{j+1} a_t$ делится на k. Поэтому мы хотели бы для каждого i найти минимальное r_i, что $\prod_{t=i}^{r_i} a_t$ делится на k. Давайте также заметим, что из тех же соображений $r_i \leqslant r_{i+1}$, поэтому нам хотелось бы применить метод двух указателей.

Чтобы применить этот метод, нам нужно разложить число k на простые множители. Тогда по каждому из них нам нужно будет найти минимальную подходящую правую границу для каждого i, и соответственно максимальная из таких границ и будет равна r_i (число делится на k, если делится на любой p^l, где p^l простой множитель в разложении числа k). Есть стандартный алгоритм разложения на простые за $O(\sqrt{k})$, но это очень долго при $k \leqslant 10^{18}$.

Давайте найдем все простые делители $\leqslant 10^6$. Для этого можно просто перебирать числа от 2 до 10^6 и делить на каждое, пока можем. Дальше у нас осталось число k'. Если $k' = 1$, то мы уже разложили число на простые множители. Если это не так, то число k либо простое, либо является произведением двух простых. Давайте сделаем массив $b_i = \gcd(a_i, k')$. Заметим, что если есть $b_i \neq 1$ и $b_i \neq k'$, то b_i — это один из простых делителей! А зная один из них, можно восстановить другой. Если же все $b_i = 1$ или $b_i = k'$, то можно считать, что число k является простым (хотя, конечно же, оно может быть и не простым, но у вас каждый элемент массива не даст множитель k' вообще, либо будет число кратное k'). Эта часть решение работает за $O(10^6 + n \cdot \log A)$, то есть за $O(n \log A)$.

Теперь когда мы знаем разложение, для каждого из простых можно решить задачу с помощью двух указателей. Количество различных простых чисел можно оценить сверху как $O(\log A)$, поэтому и эта часть решение работает за $O(n \log A)$. Общая сложность $O(n \log A)$.

Задача N. Три брата

Автор задачи: Евгений Карпович, разработчики: Евгений Карпович, Савелий Григорьев

Будем жадно моделировать процесс. Пока хотя бы одна вещь может упасть на одну клетку вниз, будем продолжать алгоритм.

Как проверить, что вещь может упасть на одну клетку вниз? Давайте для этого будем находить «заблокированные» вещи, которые уже не могут опуститься вниз. У этих вещей есть либо клетка в последней строке, либо клетка, под которой находится заблокированная вещь. Сделаем массив $can_i = 1$, если вещь незаблокированная, и 0 иначе. На каждой итерации будем находить заблокированные вещи за проход по матрице, пока такие вещи находятся. Дальше все вещи i, для которых $can_i = 1$, нужно жадно опустить на одну клетку вниз.

Оценим время работы. Мы можем моделировать процесс не более $n \cdot m$ раз (количество клеток). В процессе моделирования, мы сначала пытаемся заблокировать вещи, которые еще до этого не заблокировали — вещей у нас максимум $n \cdot m$ и поиск каждой работает за размер матрицы, то есть

$n \cdot m$. Давайте заметим, что последнее мы делаем суммарно за $O(n^2 \cdot m^2)$ по всем итерациям. Дальше мы будем не более $n \cdot m$ клеток опускать вниз. Итого $O(n^2 \cdot m^2)$.