Разбор задачи «Кот Гусь и случайная матрица»

Чтобы решить **первую подзадачу**, достаточно реализовать любое решение за $O((nm)^2)$, например, перебрать координаты двух углов, и посчитать с помощью частичных сумм сумму на подматрице.

Чтобы решить **вторую подзадачу**, необходимо уже реализовать решение за $O(n^2m)$, перебрав первую и последнюю строку подматрицы, и, таким образом, свести подзадачу к подзадаче с $n = 1$.

Чтобы окончательно решить вторую подзадачу, необходимо придумать, как решить **третью подзадачу**. На одной прямой (вертикальной или горизонтальной) необходимо найти подотрезок с максимальной суммой, делящейся на p. Это можно сделать, посчитав префиксные суммы, для каждого остатка по модулю p, а затем рассмотреть наибольшую и наименьшую суммы с таким остатком.

Таким образом, первые три подзадачи возможно решить без использования случайности входных данных.

Чтобы решить **четвертую подзадачу**, нужно воспользоваться случайностью входных данных, и реализовать более медленно правильное решение, не будем на этом останавливаться и сразу перейдем к решению, набирающему 100 баллов.

Будем строить подматрицы в порядке убывания суммы, от большей к меньшей. Для этого изначально возьмем всю матрицу полностью. Очевидно, она является подматрицой с наибольшей суммой. Какой может быть вторая по сумме матрица? Конечно, это вся матрица, из которой удалили какую-либо крайнюю строку/столбец. Аналогично, можно развить эту идею, поддерживая подматрицы в set. Каждый раз, можем доставать матрицу, которая является максимальной на данный момент, и обрезать у нее по одному строке и столбцу. Как только мы нашли матрицу с суммой делящейся на p, стоит вывести ее как ответ.

Если реализовать это с помощью set, то решение может набирать **80-100 баллов**.

Чтобы получить **100 баллов**, нужно реализовать это с помощь priority queue, не используя проверку, что такую матрицу уже рассматривали, а поддерживая указатель ptr на текущую границу, которую требуется отрезать, и сначала увеличивать x_1, затем, возможно, сдвигая указатель, затем уменьшать x_2, если указатель сейчас стоит на этой координате, и так далее.

Таким образом путь до каждой матрицы будет ровно один — последовательно подставляющий нужные координаты по очереди. В итоге, в priority queue будет храниться $(x_1, x_2, y_1, y_2, ptr)$, где ptr принимает значения 0..3.

Можно доказать, что с очень высокой вероятностью, количество матриц, которые придется рассмотреть, достаточно мало, а именно, порядка $O(p)$.

Также возможны решения, которые перебирают матрицы только «достаточно» большого размера.