Счёт в теннисе

Будем решать задачу методом динамического программирование. Причем, будем искать ответ с конца. То есть, вычислять путь минимальной длины от (a, b) до $(0, 0)$, а не наоборот. Для пересчёта можно использовать формулу $dp_{a,b} = \min(dp_{a-1,b}, dp_{a,b-1}) + \gcd(a, b)$. Однако, это решение пока работает за $O(a \cdot b)$. Попробуем его ускорить. Не умаляя общности, $a \leqslant b$. Пусть $a < b$ и b — простое. Тогда вес минимального пути от $(0, 0)$ до (a, b) равен $a + b$. Понятно, что меньше он быть не может, потому что после каждого хода вес увеличивается хотя бы на 1. А для того, чтобы его достичь, можно действовать по следующему алгоритму:

- $(0, 0)$
- $(1, 0)$
- $(1, x)$, где x увеличивается от 1 до b. При этом, после каждого изменения НОД равен 1, так как одно из чисел равно 1
- (y, b), где y увеличивается от 2 до a. При этом, после каждого изменения НОД равен 1, так как $y < b$ и b — простое

Также, если $a = 0$, от $(0, 0)$ до $(0, b)$ существует единственный путь, и его вес равен $\sum\limits_{x=1}^{b} x = \frac{(b+1) \cdot b}{2}$.

Будем вместо стандартного динамического программирования реализовывать перебор с запоминанием, который ещё называют ленивым динамическим программированием. Если при таких a и b сделать отсечение и сразу вернуть ответ за $O(1)$, решение будет работать примерно за $O(a \cdot \log(b))$. Это следует из того, что простые числа в среднем встречаются раз в логарифм чисел. И на практике нет большого разрыва между соседними простыми числами до 10^9.

Осталось научиться отсекать перебор по a. По аналогии, хотелось бы сделать это, когда a стало простым числом. Однако, возможна ситуация, при которой a — простое, $a < b$, но вес минимального пути от $(0, 0)$ до (a, b) не равен $a + b$. Поэтому, нужно добавить дополнительное условие. Пусть p — максимальное простое число, что $p \leqslant b$. Тогда, если дополнительно $a < p$ и на отрезке $[p, b]$ нет чисел делящихся на a, вес минимального пути от $(0, 0)$ до (a, b) точно равен $a + b$. Для его достижения можно действовать по следующему алгоритму:

- $(0, 0)$
- $(1, 0)$
- $(1, x)$, где $x \in [1, p]$
- (y, p), где $y \in [2, a]$
- (a, z), где $z \in [p+1, b]$. При этом, после каждого изменения НОД равен 1, так как a — простое, $a < z$ и z не делится на a

Таким образом, мы не всегда отсекаемся, когда a первый раз стало простым, но на одном из ближайших простых мы точно отсечёмся. На практике, такой перебор с отсечениями посетит максимум $\sim 10^5$ различных состояний, при ограничениях до 10^9.