**Маршруты на клетчатом поле**

Рассмотрим следующую задачу. Дана прямоугольная доска из n строк и m столбцов. В левом верхнем углу этой доски находится фишка, которую необходимо переместить в правый нижний угол. За один ход фишку разрешается передвинуть на одну клетку вниз или одну клетку вправо. Необходимо определить, сколько существует различных маршрутов фишки, ведущих из левого верхнего в правый нижний угол.

Будем считать, что положение фишки задаётся парой чисел (i,j), где i — номер строки, а j — номер столбца. Строки нумеруются сверху вниз от 0 до n−1, а столбцы — слева направо от 0 до m−1. Таким образом, первоначальное положение фишки — клетка (0,0), а конечное — клетка (n−1,m−1).

Пусть w(i,j) — количество маршрутов, ведущих в клетку (i,j) из начальной клетки. Запишем рекуррентное соотношение. В клетку (i,j) можно прийти двумя способами: из клетки (i,j−1), расположенной слева, и из клетки (i−1,j), расположенной сверху от данной. Поэтому количество маршрутов, ведущих в клетку (i,j), равно сумме количеств маршрутов, ведущих в клетку слева и сверху от неё. Получили рекуррентное соотношение:

w(i,j)=w(i,j−1)+w(i−1,j)

Это соотношение верно при i>0i>0 и j>0j>0. Зададим начальные значения: если i=0, то клетка расположена на верхнем краю доски и прийти в неё можно единственным способом — двигаясь только вправо, поэтому w(0,j)=1 для всех 0≤j<m. Аналогично, w(i,0)=1 для всех 0≤i<n.

Создадим массив w для хранения значений функции, заполним первую строку и первый столбец единицами, а затем заполним все остальные элементы массива, используя рекуррентную формулу. Поскольку каждый элемент равен сумме значений, стоящих слева и сверху, заполнять массив w будем по строкам сверху вниз, а каждую строку — слева направо.

В результате такого заполнения получим следующий массив (пример для n=4, m=5):

1 1 1 1 1

1 2 3 4 5

1 3 6 10 15

1 4 10 20 35

Код на языке Python, решающий эту задачу, будет выглядеть следующим образом:

n, m = map(int, input().split())

w = [[1] \* m **for** i **in** range(n)]

**for** i **in** range(1, n):

**for** j **in** range(1, m):

w[i][j] = w[i][j - 1] + w[i - 1][j]

**print**(w[-1][-1])

**Маршруты на клетчатом поле с дополнительными ограничениями**

Теперь решим эту задачу с дополнительным ограничением — некоторые клетки таблицы запрещены для посещения фишкой. В этой задаче нам дополнительно даётся таблица t размера n на m состоящая из 0 и 1. Если t(i,j)=0, то в клетку (i,j) перемещать фишку запрещено. Гарантируется, что t(0,0)=1.

Будем решать эту задачу аналогично предыдущей. Изменение состоит в том, что есть клетки, в которые мы не можем перемещать фишку. Формально можно записать, что если t(i,j)=0t(i,j)=0, то w(i,j)=0w(i,j)=0. Если же t(i,j)=1t(i,j)=1, то w(i,j)=w(i,j−1)+w(i−1,j). Можно объединить эти два случая в одну формулу следующим образом:

w(i,j)=t(i,j)⋅(w(i,j−1)+w(i−1,j))

Также следует более внимательно заполнить первый столбец и первую строку таблицы w, так как теперь не во всех клетках первого столбца и первой строки обязательно будут стоять единицы.

Код, решающий эту задачу, будет выглядеть следующим образом:

n, m = map(int, input().split())

t = [list(map(int, input().split())) **for** i **in** range(n)]

w = [[1] \* m **for** i **in** range(n)]

**for** i **in** range(1, n):

w[i][0] = t[i][0] \* w[i - 1][0]

**for** j **in** range(1, m):

w[0][j] = t[0][j] \* w[0][j - 1]

**for** i **in** range(1, n):

**for** j **in** range(1, m):

w[i][j] = t[i][j] \* (w[i][j - 1] + w[i - 1][j])

**print**(w[-1][-1])