**ЛАБОРАТОРНАЯ РАБОТА №4.2**

[**ДАВУМЕРНОЕ ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab11.html#id12)**:**

**ШАХМАТНЫЕ ЗАДАЧИ**

***Цель*:** научиться решать задачи двумерного динамического программирования

**Отчет по работе** должен содержать **2 файла: 1) код** программы**; 2)** файл в формате **.pdf** со скринами результатов работы (входные данные и полученный результат).

Динамическое программирование — решение сложной задачи разбиением её на более простые подзадачи, при этом каждая подзадача решается только один раз.

Динамическое программирование очень похоже на рекурсию, при этом:

* восходящее динамическое программирование — это, по сути, рекурсия с кешированием;
* нисходящее динамическое программирование — это переформулирование задачи в виде индуктивной последовательности подзадач, от крайнего случая к более сложным.

***Задача про шахматного короля, который живет на шахматной доске NxM***

Шахматный король живет на шахматной доске размером N\*M и может шагать только вправо-вниз на одну клетку (вправо на одну клетку, вниз на одну клетку и по диагонали вправо-вниз на одну клетку).

[**Задание№1**](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab11.html#id18)

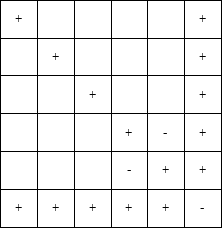
Решите задачу о количестве способов достичь клетки (N, M) из точки **1.**

[***Игра с ферзём***](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab11.html#id22) ***(по желанию)***

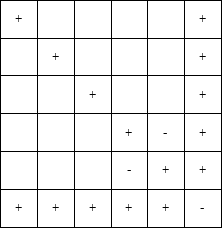
Рассмотрим игру «Ферзя в угол» для двух игроков. В левом верхнем углу доски размером N\*M находится ферзь, который может двигаться только вправо-вниз. Игроки по очереди двигают ферзя, то есть за один ход игрок может переместить ферзя либо по вертикали вниз, либо по горизонтали вправо, либо по диагонали вправо-вниз. Выигрывает игрок, который поставит ферзя в правый нижний угол. Необходимо определить, какой из игроков может выиграть в этой игре независимо от ходов другого игрока (имеет выигрышную стратегию).

Будем заполнять доску знаками «+» и «-». Знак «+» будет означать, что данная клетка является выигрышной для ходящего с неё игрока (то есть если ферзь стоит в этой клетке, то игрок, который делает ход, может всегда выиграть), а знак «-» означает, что он проигрывает. Клетки последней строки, последнего столбца и диагонали, ведущей из правого нижнего угла необходимо отметить, как «+», так как если ферзь стоит в этой клетке, то ходящий игрок может выиграть одним ходом.

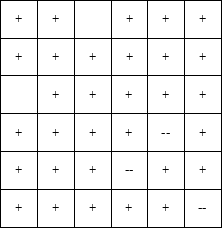
Но в правом нижнем углу необходимо поставить знак «-» — если ферзь стоит в углу, то тот игрок, который должен делать ход, уже проиграл.



Теперь рассмотрим две клетки, из которых можно пойти только в те клетки, в которых записан знак «+». В этих клетках нужно записать знак «-» — если ферзь стоит в этих клетках, то какой бы ход не сделал ходящий игрок, ферзь окажется в клетке, в которой стоит знак «+», то есть выигрывает ходящий игрок. Значит, тот, кто сейчас ходит — всегда проигрывает.

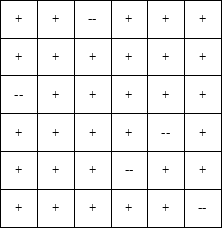


Но теперь в те клетки, из которых можно попасть в клетку, в которой стоит знак «-» за один ход, необходимо записать знак «+» — если ферзь стоит в этой клетке, то игрок, который делает ход, может выиграть, если передвинет ферзя в клетку, в которой стоит знак «-»:



Дальше таблица заполняется аналогично. В клетке ставиться знак «+», если есть ход, который ведет в клетку, в которой стоит знак «--». В клетке ставится знак «-», если все ходы из этой клетки ведут в клетки, в которых записан знак «+».

Продолжая таким образом, можно определить выигрывающего игрока для любой начальной клетки.



[**Задание №**](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab11.html#id23)**2**

Реализовать алгоритм поиска выигрышных и проигрышных позиций в игре с ферзём на прямоугольном поле M на N, где N — высота, а M — ширина поля.