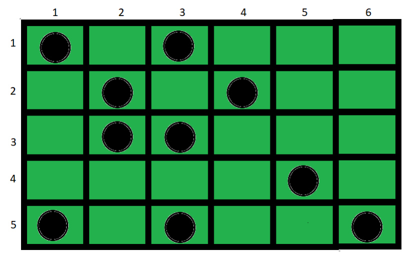
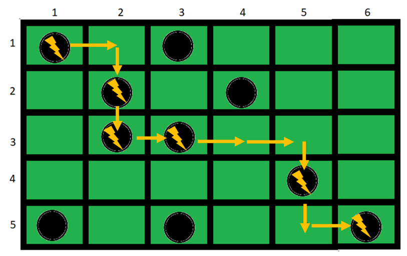
Assignment 4 is due Sunday, December 11, 23:30.

Agricultural robotics problem Consider a farm of a rectangular shape, that is divided into $n \times m$ square blocks. Some of these blocks contain weeds (i.e., unwanted plants that grow in-between crops). For instance, the left figure below illustrates a farm of size 5×6, where the blocks that contain weeds are denoted by black circles.





Consider an agricultural mobile robot that is initially located at block $(1, 1)$. The robot can navigate one block at a time, either to the right or to down. When the robot is at a block, it removes all the weeds in it.

Given the information about which blocks contain weeds, the goal is for the robot to remove weeds from as many blocks as possible, and reach the charging area located at block (n, m). For instance, for the farm shown above, the robot can remove weeds from 6 blocks as illustrated in the right figure.

Describe a dynamic programming algorithm to find a path that the robot can follow to remove weeds from the maximum number of blocks.

Submit Report (PDF file) Write a report (using an editor) including the following:

(a) **Recursive formulation** of this agricultural robotics problem.

(b) **Pseudocode of your algorithm** designed using dynamic programming based on the recursive formulation.

(c) **Asymptotic time and space complexity analysis** of your algorithm.

(d) Experimental evaluations of your algorithm: **plot** the results in a graph, and **discuss** the results (e.g., are they expected or surprising? why?)

Submit Python Code, and Benchmarks (ZIP file)

(a) Implement in Python your algorithm designed using dynamic programming, to solve the agricultural robotics problem.

(b) Create a benchmark suite of at least 5 instances to test the correctness of your Python program: take into account the **functional testing** methods (e.g., white box and black box testing) while constructing the instances, and test your programs with these instances.

(c) Create a benchmark suite to **test the performance** of your Python programs: construct instances of different sizes (e.g., relative to the number of levels), evaluate the performance of your program in terms of computation times, and plot the results within a graph.

Demos Demonstrate that your Python program correctly computes solutions for your benchmark instances, and for the instances that will be provided by us. Demo day will be announced.