Homework 4

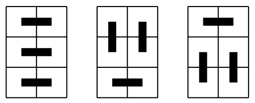
CS 331H

Due Wednesday, February 8

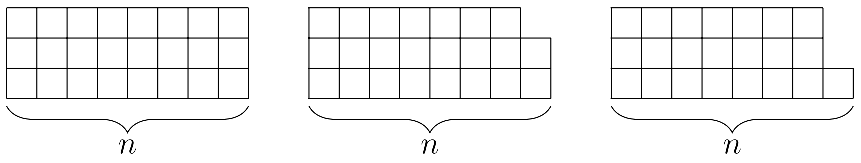
1. See the Jupyter notebook on the website.

2. Suppose you have an $n \times n$ matrix M of numbers where each entry can be either positive or negative. You want to move from the top left corner to the bottom right corner in a way so that the sum of the numbers that you traverse is as large as possible. You are allowed to move left, right, or down (but not up) and you can never visit the same square twice. Give an $O(n^2)$ time dynamic programming algorithm that finds the value of the best path from top left to bottom right.

3. (Optional) How many ways can one tile a $3 \times n$ rectangle using 2×1 tiles? For example, there are 3 ways to tile a 3×2 rectangle:



   (a) Show how to compute the answer in $O(n)$ time, assuming that the word size can represent numbers as large as the answer. You may find it useful to consider the number of ways to tile all of the following figures:



   (b) By expressing the recursion in terms of matrices, show how to compute the part (a) answer in $O(\log n)$ time, again assuming the word size can represent numbers as large as the answer.

   (c) Now suppose that the word size is $\Theta(\log n)$. How much time do your solutions take?